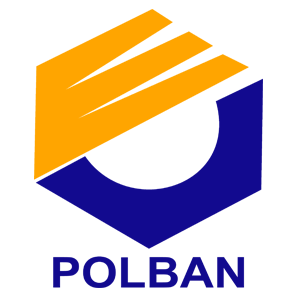
**LAPORAN DASAR-DASAR PEMROGRAMAN**

**APLIKASI PERMAINAN CONGKLAK**

Untuk Memenuhi Salah Satu Tugas

Mata Kuliah Dasar-Dasar Pemrograman

Dosen : Ani Rahmani



Disusun oleh

Anggita Intan Fania (171524004)

Hamzah Prasetyo Utomo (171524011)

Ilham Gibran Achmad Mudzakir (171524012)

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

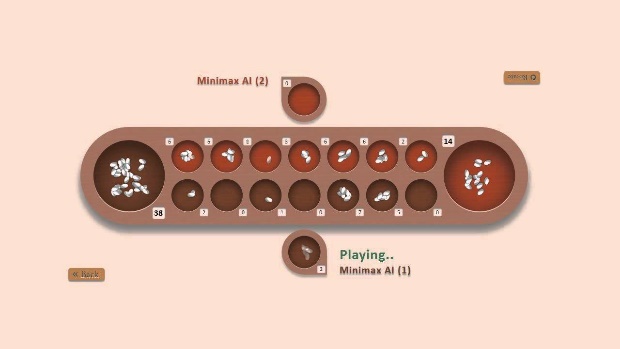
**PRODI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2018**

# BAB I DESKRIPSI UMUM

## Permainan Congklak Secara Umum

Congklak adalah permainan yang dimainkan oleh 2 orang pemain biasanya dimainkan oleh perempuan dengan menggunakan papan congklak yang terdapat 7 lubang kecil pemain, 7 lubang kecil lawan, 1 lubang besar milik sendiri dan 1 lubang besar milik musuh dimana lubang besar ini disebut lumbung (lihat Gambar 1), serta biji congklak (lihat Gambar 2). Inti dari permainan ini adalah mengumpulkan biji congklak di lubang pojok yang menjadi milik kita. Menang atau kalah ditentukan dari banyaknya biji yang berhasil dikumpulkan, yang paling banyak maka dia pemenangnya.

*Gambar 1 papan dakon (Sumber: google image)*

*Gambar 2 biji congklak (Sumber: google image)*

Terdapat beberapa aturan dalam permainan congklak, yaitu :

1. Terdapat beberapa 2 versi dalam memulai permainan congklak, yaitu :

* Pemain melakukan suit terlebih dahulu untuk menentukan pemain pertama.
* Pemain dan lawan bermain secara bersama dan jika yang pertama mati harus menunggu pemain yang belum mati untuk berganti giliran main.

1. Jika biji terakhir pemain masuk ke lubang yang terdapat bijinya, maka pemain dapat melanjutkan permainan sampai biji yang ada di tangan masuk ke dalam lubang yang kosong baik di lubang sendiri atau lawan, pemain dinyatakan mati dan digantikan gilirannya oleh lawan.
2. Pemain dilarang mengisi lumbung lawan dengan biji.
3. Jika biji terakhir jatuh pada lubang besar milik sendiri maka pemain dapat memilih kembali seluruh biji yang terdapat dalam salah satu lubang kecil milik pemain.
4. Pemain dapat mengambil biji lawan dan biji terakhir yang sejajar dengan lubang tempat biji terakhir berhenti, dengan syarat biji berhenti di lubang milik sendiri dan lubang milik lawan tidak kosong.
5. Permainan akan terus berlanjut sampai syarat permainan berakhir terpenuhi, terdapat dua versi syarat berakhirnya permainan, yaitu :

* Ketika lubang kecil pemain atau lawan telah habis/kosong. Meskipun lubang kecil pemain/lawan masih terdapat biji. Pada versi ini biji yang masih ada dalam lubang kecil secara otomatis dihitung ke dalam lumbung pemilik biji.
* Ketika lubang kecil pemain maupun lawan telah habis/kosong. Biji yang tersisa hanya di lumbung milik masing-masing.

1. Pemain dapat memilih untuk melanjutkan permainan dengan mengisi kembali lubang kecil masing-masing 7 biji dengan biji yang telah dikumpulkan. Jika biji tidak mencukupi maka pemain akan mendapatkan lubang yang kosong. Lubang tersebut dilarang dilewati oleh para pemain, biasanya diberikan tanda agar pemain tidak melewati lubang tersebut. Permainan berakhir jika syarat berakhirnya permainan terpenuhi dan pemain tidak melanjutkan lagi bermain.

# BAB II DESKRIPSI APLIKASI

## Algoritma *Greedy*

*Greedy* merupakan algoritma yang digunakan untuk menyelesaikan masalah secara langkah per langkah (*step-by-step*). Setiap langkahnya terdapat opsi yang dapat dipilih. Algoritma *greedy* memiliki konsep memilih pilihan terbaik pada setiap langkahnya (optimum lokal). Namun hal ini tidak berarti solusi yang dipilih oleh *greedy* merupakan pilihan terbaik. Karena pada kenyataannya, algoritma *greedy* tidak beroperasi secara menyeluruh terhadap alternatif solusi yang ada sehingga belum tentu algoritma *greedy* merupakan optimum global.

Algoritma *greedy* disusun oleh 5 elemen, yaitu :

1. Himpunan Kandidat

Berisi opsi-opsi yang dapat dipilih pada setiap pilihan.

1. Himpunan Solusi

Kumpulan solusi yang dihasilkan pada setiap langkah.himpunan ini berisi elemen-elemen dari himpunan kandidat yang terpilih.

1. Fungsi seleksi

Fungsi seleksi merupakan cara untuk melakukan pemilihan opsi sehingga menjadi solusi.

1. Fungsi kelayakan

Fungsi kelayakan adalah batasan-batasan dalam membentuk solusi dan menjadi tolak ukur apakah pilha tersebut dapat dipilih.

1. Fungsi objektif

Bertujuan untuk memaksimalkan atau meminimunkan solusi yang dipilih.

## Implementasi Aturan Permainan pada Program

Dari peraturan yang ada, pada umumnya peraturan pada program yang kami buat tidak berbeda jauh dengan peraturan yang telah ada, yaitu :

1. Pada awal permainan, pemain melakukan suit dengan komputer. Sehingga dapat dilihat siapa yang akan memulai permainan
2. Pemain yang tidak mendapatkan giliran menunggu sampai pemain yang sedang bermain mati.
3. Pemain dapat melanjutkan bermain apabila :

* Biji terakhir pemain jatuh pada lubang yang terdapat biji/tidak kosong baik itu di daerah lawan atau daerah pemain.
* Biji terakhir pemain jatuh di lumbung milik pemain, maka pemain dapat memilih kembali lubang kecil milik pemain yang berisi biji.

1. Pemain dikatakan mati jika biji terakhir jatuh pada lubang kecil kosong. Terdapat 2 kemungkinan saat pemain mati, yaitu :

* Jika pemain mati di daerahnya dan lubang di daerah lawan yang sejajar dengan lubang tempat pemain mati, maka pemain melakukan tembak dengan mengambil biji lawan dan biji terakhirnya dan menyimpannya ke lumbung milik sendiri. Jika di lubang di daerah lawan kosong maka pemain tidak melakukan tembak.
* Jika pemain mati di daerah lawan maka giliran pemain digantikan oleh lawan.

1. Permainan akan berakhir jika sudah tidak terdapat biji di lubang kecil di daerah lawan maupun daerah pemain.
2. Untuk bermain kembali, pemain harus memulai permainan dari awal.

## Permainan Utama

### **Analisis Arena Permainan**

Arena permainan pada program permainan congklak ini terdapat 3 daerah, yaitu :

1. Daerah pemain, yaitu daerah permainan pemain yang terdiri dari 7 buah lubang. Setiap lubang akan terisi 7 buah biji pada saat awal permainan. Daerah ini digunakan pemain untuk pemilihan langkah permainan dan melakukan tembak. Dalam program yang kami buat untuk daerah pemain lubang diberi nilai dari 1-7.
2. Daerah Computer, yaitu daerah permainan Computer yang terdiri dari 7 buah lubang. Setiap lubang akan terisi 7 buah biji pada saat awal permainan. Daerah ini digunakan Computer untuk pemilihan langkah permainan dan melakukan tembak. Proses ini nantinya akan menggunakan algoritma *Greedy* untuk melakukan pemilihan tersebut. Dalam program yang kami buat untuk daerah Computer lubang diberi nilai 9-15..
3. Lumbung, yaitu daerah lumbung pemain di sebelah kiri pemain dan lumbung Computer di sebelah kanan pemain. Lumbung ini tempat menyimpan biji yang terkumpul. Pada program yang kami buat lumbung pemain diberi nilai 8 dan untuk lumbung Computer diberi nilai 16.

### **Penentuan Pemain dan Giliran**

Terdapat 2 pemain dalam permainan congklak ini, yaitu :

1. *User* sebagai pemain yang memainkan secara langsung melalui komputer. Di awal permainan *user* akan ditanyakan nama untuk data *highscore.*
2. Pemain yang lainnya adalah Computer.

Untuk giliran, pada awal permainan akan dilakukan suit dan untuk suit Computer dilakukan secara random sehingga *user* belum pasti bermain pertama. Pada program yang kami buat, giliran dapat dilihat pada bagian tengah atas, ketika giliran pemain akan menunjukan giliran pemain dan sebaliknya.

### **Alur Permainan dalam Satu Kali Giliran**

Berikut ini adalah hal-hal yang dilakukan saat giliran pemain :

1. Memilih lubang yang akan dimainkan.
2. Jika giliran bermain adalah *user* maka *user* dapat memilih lubang yang akan dimainkan. Dengan catatan lubang di daerah pemain yang dapat dipilih.
3. Jika giliran bermain adalah Computer maka Computer akan memilih lubang dengan metode *greedy.*
4. Mendistribusikan satu per satu biji di tangan sampai habis. Pada program yang kami buat dalam mendistribusikan biji hanya terlihat angka bertambah dan pada bagian tengah papan terdapat angka sebagai ciri biji di tangan dan ketika biji didistribusikan, angka tersebut akan berkurang. Pada program, ketika mendistribusikan lubang yang sedang ditempati akan berwarna merah terang.
5. Jika biji terakhir berakhir pada lubang yang tidak kosong, pemain melanjutkan bermain.
6. Jika biji terakhir berakhir pada lubang yang kosong, giliran bermain akan digantikan oleh lawan. Terdapat 2 kemungkinan jika pemain mati, yaitu
7. Jika mati di daerah sendiri dan lubang daerah lawan yang sejajar tidak kosong, pemain dapat melakukan tembak.
8. Jika mati di daerah lawan atau mati di daerah sendiri tetapi lubang daerah lawan yang sejajar kosong, pemain tidak dapat melakukan tembak.

### Analisis algoritma Greedy

Seperti yang telah dijelaskan sebelumnya algoritma *greedy* akan digunakan oleh Computer untuk membangun jalannya permainan.

Dalam algoritma *greedy* terdapat beberapa elemen-elemen yang perlu dipertimbangkan yaitu himpunan kandidat, himpunan solusi, himpunan fungsi seleksi, fungsi kelayakan, fungsi objektif. Berikut adalah penjelasan dari elemen-elemen tersebut.

1. Himpunan Kandidat

Himpunan kandidat pada permainan congklak ini adalah lubang-lubang kecil. Namun karena lubang kosong tidak dapat dipilih maka, himpunan kandidatnya adalah lubang kecil yang berisi biji.

1. Himpunan Solusi

Lubang yang dipilih untuk dijalankan. Karena langkah yang optimal bergantung pada gerakan lawan dan sebaliknya maka untuk himpunan solusi dalam permainan tidak pasti.

1. Fungsi Seleksi

Fungsi seleksi merupakan fungsi untuk memilih lubang yang akan dijalankan mencapai solusi yang optimum. Dalam permainan congklak yang kami buat dibuat terdapat 2 fungsi seleksi yaitu memilih lubang yang dapat masuk tepat ke lumbung dan memilih lubang dengan biji terbanyak agar tidak ditembak lawan.

1. Fungsi Kelayakan

Lubang yang harus dipilih adalah lubang di daerah pemain bukan lawan. Lubang hanya dapat dipilih satu lubang dalam setiap satu kali jalan.

1. Fungi Objektif

Langkah yang diambil jumlahnya mnimum.

### Penentuan Pemenang

Jika daerah pemain kosong tetapi daerah lawan masih terdapat biji maka lawan akan terus mendapatkan giliran sampai daerah lawan juga kosong dan sebaliknya. Jika kedua daerah telah kosong maka permainan berakhir dan akan ditentukan pemenangnya.

### Penyimpanan *Highscore*

Setelah selesai permainan, data *user* menang atau kalah akan disimpan dengan banyaknya biji terkumpul, banyak menembak, banyak giliran. Kemudian dikirimkan ke *Highscore*.

## Highscore

Dalam data *highscore* kami menentukan kriteria yang dapat ditentukan untuk menentukan prestasi pemain, yaitu :

1. Banyak biji terkumpul, semakin banyak biji terkumpul semakin baik karena tujuan dari permainan ini adalah mendapatkan biji sebanyak-banyaknya.
2. Banyak menembak, semakin banyak menembak semakin baik karena akan mendapatkan biji dengan cepat.
3. Banyak giliran, semakin sedikit giliran semakin baik karena bersifat kompetitif terhadap pemain lawan.
4. Status, apakah *user* menang atau kalah.

Dalam fitur *highscore* iniakan mencatat pencapaian *user* dan menyimpannya sehingga dapat dilihat kembali rekamannya oleh *user*. Kami membuat fitur ini agar *user* bebasuntuk memilih sesuai dengan kriterianya.

## **Tentang Aplikasi dan Bantuan**

Dalam program ini terdapat fitur tentang aplikasi dan bantuan, pada fitur tentang aplikasi akan ditampilkan teks dari suatu *file* yang berisi tentang aplikasi permainan congklak secara umum. Fitur bantuan berisi tentang penjelasan secara singkat mengenai cara bermain permainan congklak.

## Menu Utama

Program akan menampilkan menu utama saat program pertama kali dijalankan yang terdiri dari :

1. Mulai Permainan
2. *Highscore*
3. Tentang Permainan
4. Bantuan
5. Keluar

# BAB III DESKRIPSI KEBUTUHAN DATA

Dalam permainan congklak/dakon, ada kebutuhan untuk menyimpan informasi dan menjawab kebutuhan data:

## Papan Congklak

* 1. Jumlah biji pada tiap lubang (14 lubang main dan 2 base pemain);
  2. Jumlah biji yang dipegang (ditunjukan ditengah papan);
  3. Nomor lubang (1 hingga 16) untuk menentukan apakah terjadinya “tembak”, bergantinya giliran dan lubang mana yang boleh dipilih oleh *user*;
  4. Apakah terdapat biji pada lubang disisi pemain (jika pada semua lubang tidak terdapat biji, maka giliran pemain tersebut dilewat).

## Pemain

* 1. Nama pemain (data highscore);
  2. Banyak pemain melakukan penembakan (data highscore);
  3. Banyak pemain mendapat giliran (data highscore);
  4. Banyak biji pada base tiap pemain;
  5. Kondisi akhir pemain (menang,kalah, atau seimbang);
  6. Dikendalikannya pemain oleh *user* atau komputer (sebagai penentu dilakukan atau tidaknya modul AI).

## Dinamika Permainan

Untuk memperlancar proses pembuatan program dan terbentuknya tata data yang rapih dan mudah dilihat, dibuatlah data tipe baru (*user-defined*) yang mampu menampung data-data yang diperlukan dalam program congklak. Namun, tentu saja tidak semua data yang digunakan memerlukan pembuatan data tipe baru, beberapa data yang digunakan dapat terakomodir cukup dengan menggunakan tipe data primitif yang telah tersedia.

## Struktur *User-Defined Types*

Informasi mengenai pemain dan data highscore merupakan data terstruktur. Untuk memudahkan penggunaan dan teraturnya data-data tersebut, dibentuklah tipe data baru (*User-Defined Types*), yaitu:

1. **pemain**, merupakan data yang digunakan sebagai penampung data pemain.
2. **highscore**, merupakan data yang digunakan sebagai penampung data highscore pada proses pembacaan dan penulisan file **score\_congklak.dat**.

Tabel 3.1 Struktur Tipe Data **pemain**

|  |  |  |
| --- | --- | --- |
| Nama Variabel | Tipe Data | Keterangan |
| Nama | Character[25] | Nama Pemain untuk disimpan dalam *highscore* |
| bijiCollected | Integer | Jumlah biji yang pemain berhasil kumpulkan dalam “base” |
| numberOfShoot | Integer | Jumlah pemain melakukan “tembak” untuk disimpan dalam *highscore* |
| numberOfTake | Integer | Jumlah pemain mendapat giliran bermain untuk disimpan dalam *highscore* |
| isComputer | Boolean | TRUE: pemain merupakan komputer (AI); False : pemain merupakan *user* |
| Status | Integer | 1: menang; 2: seimbang; dan 3: kalah |

Tabel 3.2 Struktur Tipe Data **highscore**

|  |  |  |
| --- | --- | --- |
| Nama Variabel | Tipe Data | Keterangan |
| Nama | Character[25] | Nama Pemain pemegang *score* |
| bijiCollected | Integer | Jumlah biji yang pemain berhasil kumpulkan dalam “base” |
| numberOfShoot | Integer | Jumlah pemain melakukan “tembak” |
| numberOfTake | Integer | Jumlah pemain mendapat giliran bermain |
| Status | Integer | 1: menang; 2: seimbang; dan 3: kalah |

## Variabel Penting

Selain data tipe baru, terdapat juga data-data tipe primitif yang berperan penting dalam berjalannya program, data tersebut antara lain:

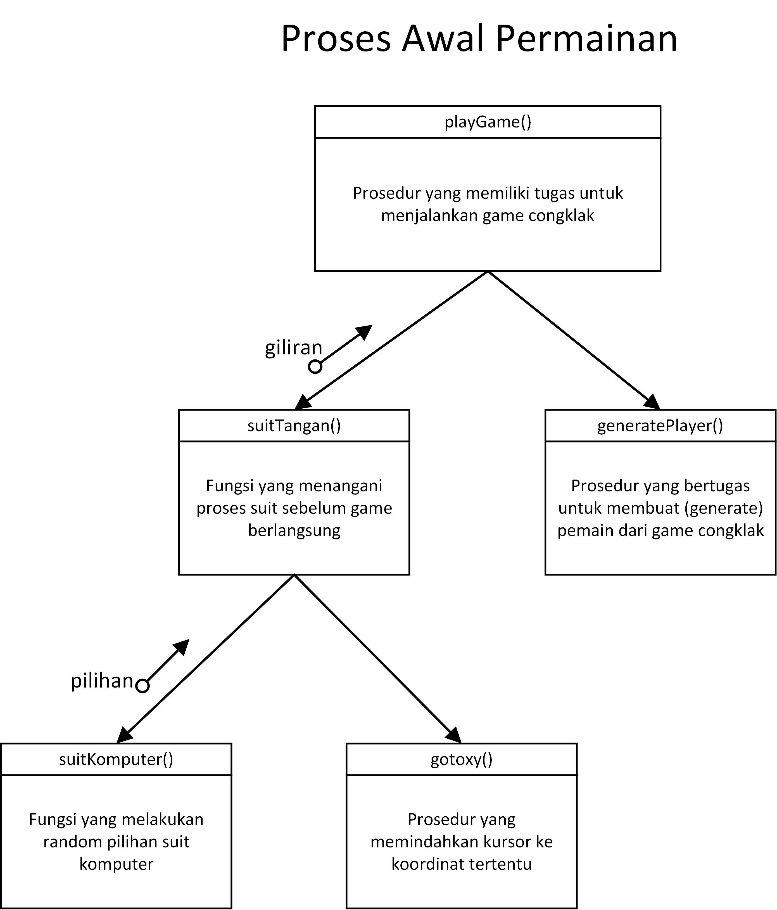
Tabel 3.3 Tabel Variabel Penting

|  |  |  |
| --- | --- | --- |
| Nama Variabel | Tipe Data | Keterangan |
| Kewuk | Integer[14] | Penampung jumlah biji pada lubang-lubang permainan kecuali *base* |
| Hand | Integer | Jumlah biji yang sedang disebarkan oleh pemain ke lubang satu persatu |
| Turn | Integer | 1: giliran *user*; 2:giliran komputer; |

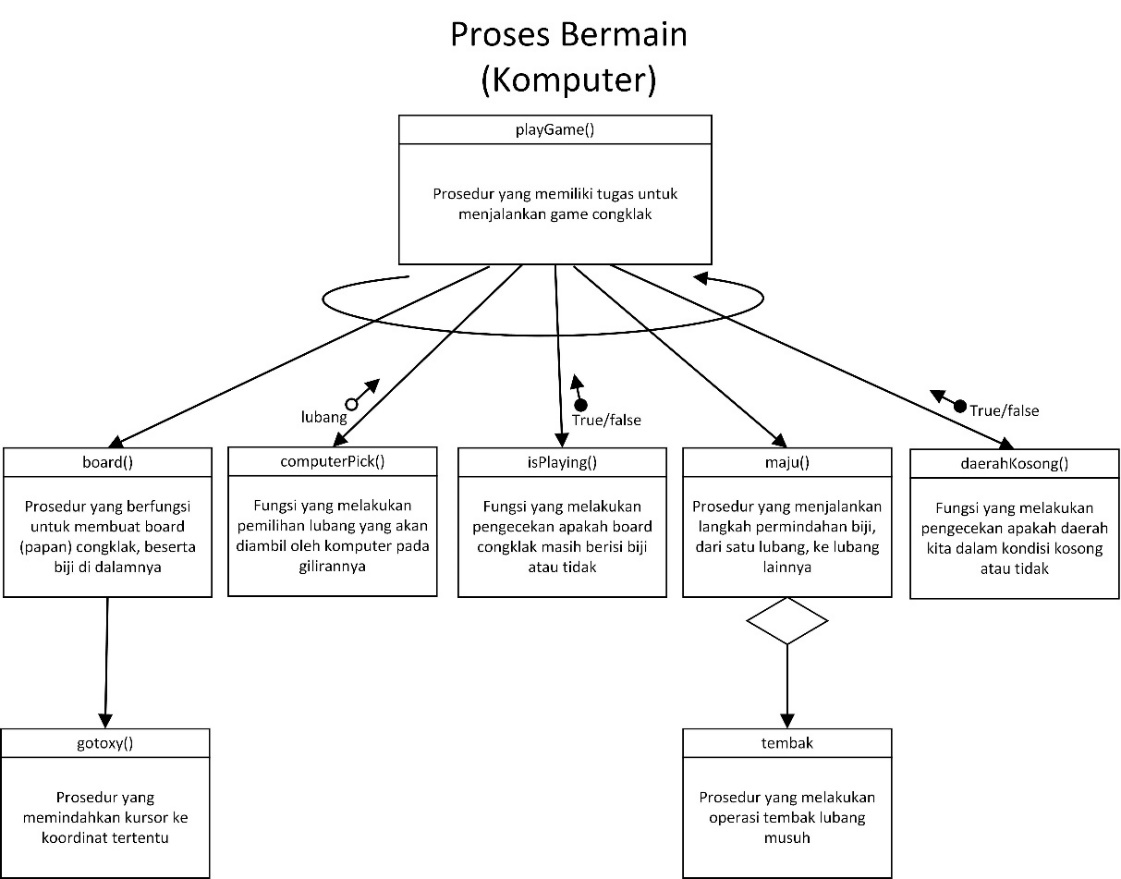
# BAB IV RANCANGAN PROSES

## Structure Chart

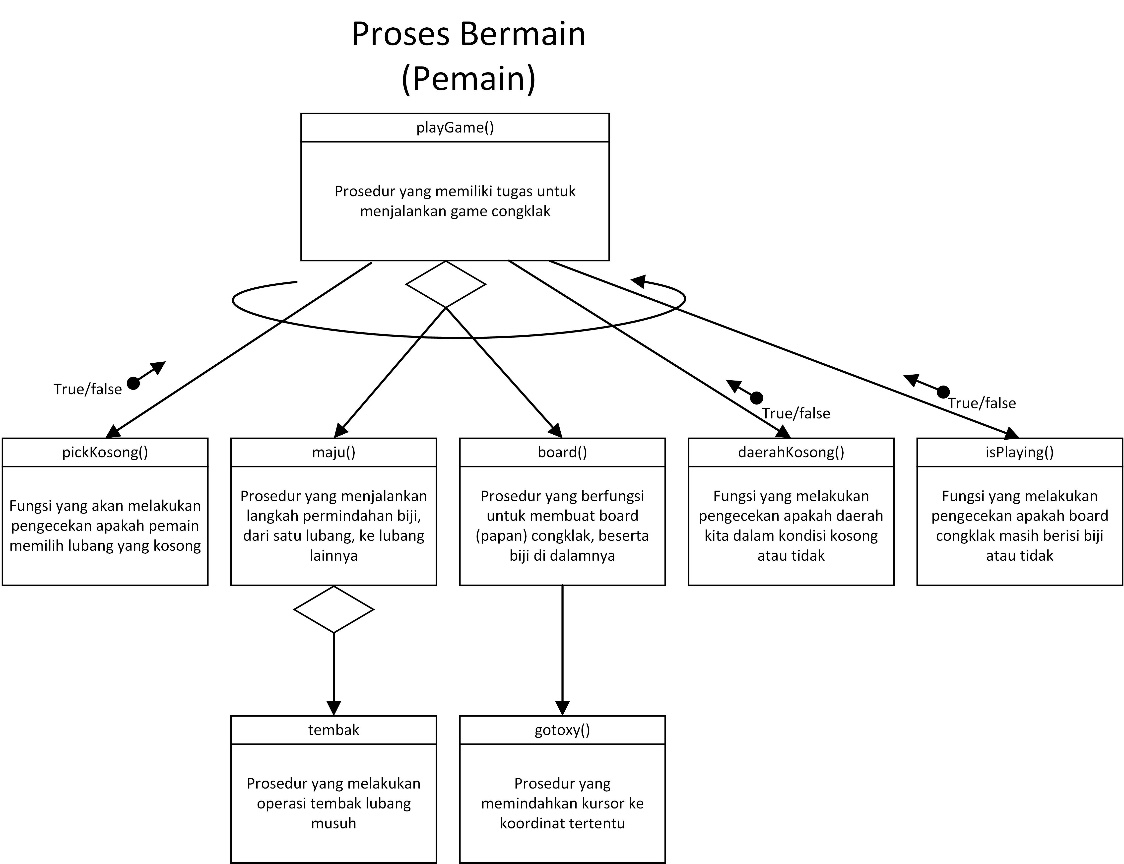
Berikut adalah rancangan *structure chart* yang kelompok kami buat dalam pembuatan aplikasi congklak menggunakan bahasa C. Terdapat lima *structure* *chart* yang dibuat dalam perancangan proses yang terdapat pada aplikasi congklak ini, yang dapat dikategorikan menjadi tiga kategori, yaitu awal permainan, saat permainan dan setelah permainan. Berikut adalah *structure chart* yang dibuat dalam perancangan aplikasi congklak.

****

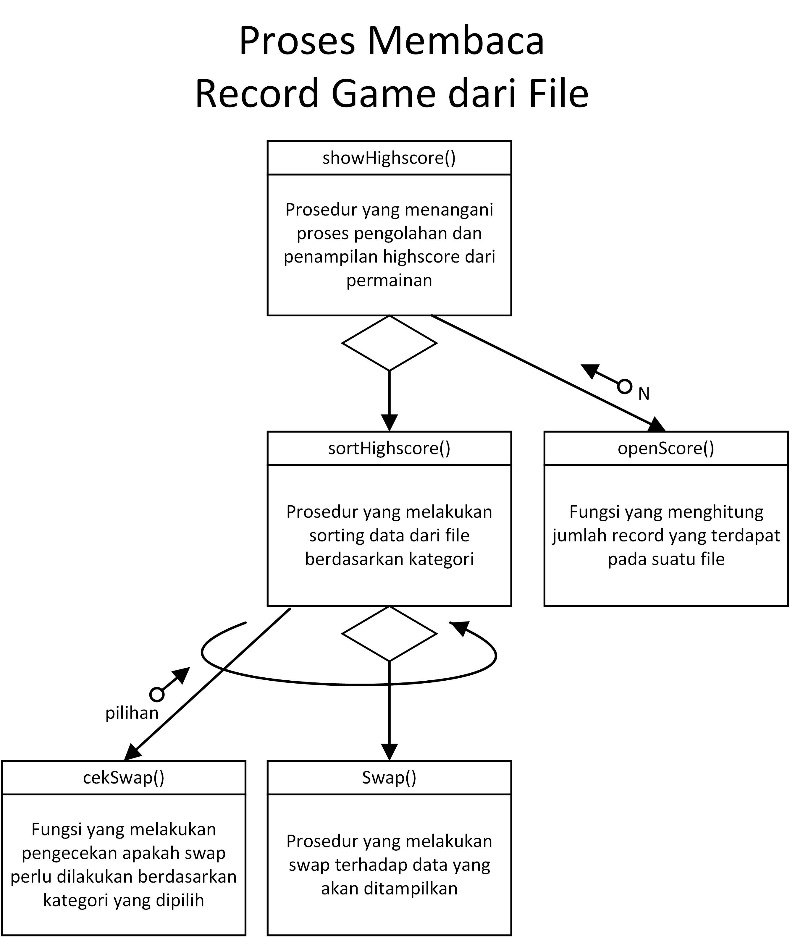
Gambar 4.1 Proses Awal Permainan



Gambar 4.2 Proses Bermain (Komputer)



Gambar 4.3 Proses Bermain (Pemain)



Gambar 4.4 Proses Membaca Data dari File



Gambar 4.5 Proses Menyimpan Data ke File

## **Program Utama**

Secara umum, program utama hanya berfungsi untuk meminta *user* memilih empat fitur yang dimiliki oleh program secara keseluruhan. Setiap satu modul selesai bekerja, maka alur proses di dalam program utama diatur agar kembali ke tampilan *main menu*, sehingga *user* bisa menjalankan modul lain.

|  |  |
| --- | --- |
| Nama Modul | main |
| Deskripsi | Program utama |
| *Author* | Hamzah Prasetio U, Ilham Gibran A.M |
| Jenis | Prosedur |
| *Initial State* | Program belum berjalan |
| *Final State* | Terlihat pilihan menu di layar, di mana *user* bisa memilih modul mana yang ingin dieksekusi |
| Kamus Data | |
| Pilihan: **integer** | |
| Algoritma | |
| **Do**  **write(layar)** pilihan menu yang tersedia  **input(pilihan)** pilihan menu dari *user*  **if** pilihan=1  **call** playGame()  **else if** pilihan=2  **call** showHighscore()  **else if** pilihan=3  **call** tentang()  **else if** pilihan=4  **call** bantuan()  **else if** pilihan=5  **write(layar)** ucapan selamat tinggal  **else**  **write(layar)** ucapan menu tidak tersedia  **endif**  **while** pilihan != 5 {agar setelah dari modul bisa ke menu utama lagi} | |

## Proses Bermain Congklak

|  |  |
| --- | --- |
| Nama Modul | playGame |
| Deskripsi | Menyiapkan data pemain, menjalankan permainan secara keseluruhan, dan mengirimkan data *user* ke *highscore* |
| *Author* | Ilham Gibran A.M, Hamzah Prasetio U, Anggita Intan F |
| Jenis | Prosedur |
| *Initial State* | Tampilan *main menu* pada layar |
| *Final State* | *User* dapat bermain Ludo hingga tampil kondisi game berakhir |
| Kamus Data | |
| nama: **char[25]**  lubang: **integer**  biji: **integer**  P: **pemain[2]**  Turn: **integer** | |
| Algoritma | |
| **Clearscreen**  **input(name)** nama *user*  **call** generatePlayer(P,name)  Turn 🡨 suitTangan()  **If** turn=0  **write(layar)** pemberitahuan komputer mendapat giliran pertama  **else**  **write(layar)** pemberitahuan*user*mendapat giliran pertama  **endif**  **do**  **if** turn=1  **if NOT**daerahKosong(turn)  P[0].numberOfTake++  board()  **input(lubang)** nomor lubang untuk digerakkan  **if** lubang<=0 OR lubang>=8 OR pickKosong(lubang)  **clearscreen**  **else**  **call** maju(lubang,turn)  **call** board(-1)  **endif**  **else**  turn 🡨 0  **endif**  **else**  **if** **NOT**daerahKosong(turn)  **clearscreen**  **call** board(-1)  **call** srand(time(NULL))  lubang 🡨 computerPick()  **write(layar)** lubang yang dipilih oleh komputer  **call** maju(lubang,turn)  **else**  turn 🡨 1  **endif**  **endif**  **while call** isplaying()  P[0].status 🡨 WinOrLose()  **Call** saveScore()  **Call** showHighscore() | |

|  |  |
| --- | --- |
| Nama Modul | suitTangan |
| Deskripsi | Menentukan siapa yang mendapat giliran pertama |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Fungsi |
| *Output* | **1 :** *user* mendapat giliran pertama  **0** : komputer mendapat giliran pertama |
| Kamus Data | |
| Opsi : **integer**  Komputer : **integer** | |
| Algoritma | |
| **do**  **if** opsi != 0  **write(layar)** input *user* salah  **endif**  **write(layar)** menu pilihan untuk suit  **input(opsi)** pilihan yang dipilih oleh *user*  **while** opsi < 1 || opsi > 5  **if** opsi!=4  komputer 🡨 suitKomputer(opsi)  **if** opsi **MOD** 3 > komputer **MOD** 3  **if** opsi = 2 **AND** komputer = 3  **return** 0  **endif**  **return** 1  **else**  **if** opsi = 3 **AND** komputer = 2  **return** 1  **endif**  **return** 0  **endif**  **else** **if** opsi = 5 {cheat jika ingin giliran kedua}  **return** 0  **else**  **return** 1 {cheat jika ingin giliran pertama}  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | suitKomputer |
| Deskripsi | Menghasilkan pilihan komputer saat proses suit secara random |
| *Author* | Ilham Gibra Achmad Mudzakir |
| Jenis | Fungsi |
| *Output* | **1-3** : angka yang mewakili gunting-kertas-batu |
| Kamus Data | |
| hasil : **integer** | |
| Algoritma | |
| **Call** srand(time(NULL))  **do**  hasil 🡨 rand() **MOD** 3 + 1  **while** hasil = pilihan  **depend on** hasil  **case** 1 : **write(layar)** Komputer Memilih Kertas  **case** 2: **write(layar)** Komputer Memilih Gunting  **case** 3: **write(layar)** Komputer Memilih Batu  **return** hasil | |

|  |  |
| --- | --- |
| Nama Modul | generatePlayer |
| Deskripsi | Mempersiapkan variabel pemain untuk permainan |
| *Author* |  |
| Jenis | Prosedur |
| Initial State | Variabel pemain masih kosong atau terisi bekas permainan sebelumnya |
| Final State | Variabel pemain dalam kondisi siap digunakan |
| *Input* | **P:** pemain[ ], variabel penampung data pemain  **Nama**: char[ ], variabel nama *user* |
| *Parameter Output* | **P:** pemain[ ], variabel penampung data pemain |
| Kamus Data | |
| **I** : integer | |
| Algoritma | |
| **For** i 🡨 0 to 1  **If** i=0  strcpy(P[i].nama,nama)  P[i].isComputer = false  P[i].bijiCollected = 0  P[i].numberOfShoot = 0  P[i].numberOfTake = 0  **else**  strcpy(P[i].nama,"Computer")  P[i].isComputer = true;  **endif**  P[i].bijiCollected = 0  **endfor** | |

|  |  |
| --- | --- |
| Nama Modul | Board |
| Deskripsi | Membuat papan congklak pada layar dan mengisi setiap kuwuk |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Prosedur |
| *Initial State* | Tampilan menu utama atau tampilan papan congklak sebelumnya |
| *Final State* | Tampil papan congklak dan isi kuwuknya |
| *input* | **Lubang:** integer |
| Kamus Data | |
| papan: **char** **papan[5][25]** ={  {'-','-','-','9','-','-','A','-','-','B','-','-','C','-','-','D','-','-','E','-','-','F','-','-','-'},  {'/',' ','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|',' ','\\'},  {'|','|','X','|',' ',' ',' ',' ',' ',' ',' ',' ','Z',' ',' ',' ',' ',' ',' ',' ',' ','|','X','|','|'},  {'\\',' ','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|','|','x','|',' ','/'},  {'-','-','-','7','-','-','6','-','-','5','-','-','4','-','-','3','-','-','2','-','-','1','-','-','-'},  }  i,j,x,y,index,number,z,posisi: **integer**  giliran: **char[9]**  Turn: **integer** | |
| Algoritma | |
| HANDLE hdl 🡨 GetStdHandle(STD\_OUTPUT\_HANDLE)  number 🡨 10  **clearscreen**  **if** lubang>8 **AND** lubang<16  posisi 🡨 lubang - 9  **else**  posisi 🡨 14-lubang  **endif**  y 🡨 2  **for** i 🡨 0 to 4  y 🡨 y + 2  x 🡨 20  **for** j 🡨 0 to 24  x 🡨 x + 3  **call** gotoxy(x,y)  **if** papan[i][j] > 64 **AND** papan[i][j] < 71  **write(layar)** number  number++  **else** **if** papan[i][j] = 'x'  **if** index = posisi **AND** lubang != 8 **AND** lubang != 16  SetConsoleTextAttribute(hdl,6)  **endif**  **write(layar)** kewuk[index]  SetConsoleTextAttribute(hdl,15)  index++  **else** **if** papan[i][j] = 'X'  **if** z != turn **AND** z = lubang/-8 +1 **AND** lubang **MOD** 8 = 0  SetConsoleTextAttribute(hdl,6)  **endif**  **write(layar)** P[z].bijiCollected  SetConsoleTextAttribute(hdl,15)  z++  **else** **if** papan[i][j] = 'Z'  **write(layar)** hand  **else**  **write(layar)** papan[i][j]  **endif**  **endfor**  **write(layar)** \n  **endfor**    **if** turn = 1  strcpy(giliran,"Pemain")  **else**  strcpy(giliran,"Komputer")  **endif**  gotoxy(50,0) **write(layar)** giliran | |

|  |  |
| --- | --- |
| Nama Modul | computerPick |
| Deskripsi | Menentukan pilihan lubang untuk komputer |
| *Author* | Ilham Gibran Achmad Mudzakir, Anggita Intan F |
| Jenis | Fungsi |
| *Output* | **9-15:** nomor lubang yang akan dipilih komputer |
| Kamus Data | |
| Pick,index,realPosition,pickPosition: **integer**  Memilih: **boolean** | |
| Algoritma | |
| Memilih 🡨 false  index 🡨 9  pick 🡨 9  **while** index < 16 **AND** !memilih  realPosition 🡨 index - 9  **if** kewuk[realPosition] + index = 16  pick 🡨 index  memilih 🡨 true  **endif**  index 🡨 index + 1;  **endwhile**  index 🡨 9  **while** index < 16 **AND** !memilih  realPosition 🡨 index - 9  pickPosition 🡨 pick - 9  **if** kewuk[realPosition] > kewuk[pickPosition]  pick = index  **endif**  index 🡨 index + 1  **endwhile**  **return** pick | |

|  |  |
| --- | --- |
| Nama Modul | isPlaying |
| Deskripsi | Memeriksa apakah permainan masih berjalan atau tidak |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Fungsi |
| *Output* | **playing:** boolean yang menyatakan masih atau tidak berjalannya permainan |
| Kamus Data | |
| I: **integer**  playing: **boolean** | |
| Algoritma | |
| playing 🡨 false  **while** i<14 **AND NOT**playing  **if** kewuk[i] != 0  playing 🡨 true  **endif**  i 🡨 i + 1  **endwhile**  **return** playing | |

|  |  |
| --- | --- |
| Nama Modul | WinOrLose |
| Deskripsi | Memeriksa siapa pemenang game |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Fungsi |
| *Output* | Nilai integer yang menunjukan status permainan bagi *user* |
| Algoritma | |
| **If** P[0].bijiCollected > P[1].bijiCollected  **return** 1  **else** **if** P[0].bijiCollected < P[1].bijiCollected  **return** 0  **else**  **return** 2  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | daerahKosong |
| Deskripsi | Memeriksa apakah deretan lubang pada bagian papan pemain yang mendapat giliran kosong atau tidak |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Fungsi |
| *Output* | **kosong:** boolean yang menyatakan kosong atau tidak kuwuk |
| Kamus Data | |
| Awal,akhir: **integer**  kosong: **boolean** | |
| Algoritma | |
| kosong 🡨 true  **depend on** giliran  **case** 0 : awal 🡨 0  akhir 🡨 6  **case** 1 : awal 🡨 7  akhir 🡨 13  **default**: awal 🡨 0  **while** awal <= akhir **AND** kosong  **if** kewuk[awal] != 0  kosong 🡨 false  **endif**  awal 🡨 awal + 1  **endwhile**  **return** kosong | |

|  |  |
| --- | --- |
| Nama Modul | Maju |
| Deskripsi | Menyebarkan biji pemain |
| *Author* | Ilham Gibran A.M, Hamzah Prasetio, Anggita Intan F |
| Jenis | Fungsi |
| *input* | Lubang**: integer**  Giliran: **integer** |
| *Iniital State* | Pemain memilih lubang yang bijinya akan disebar |
| *Final State* | Biji dari lubang tersebut telah disebar |
| Kamus Data | |
| realPosition,i,player: **integer**  out: **boolean** | |
| Algoritma | |
| Player 🡨 giliran  Out 🡨 true  **do**  **if** lubang>8 **AND** lubang<16  realPosition 🡨 lubang - 9  **else**  realPosition 🡨 14-lubang  **endif**  hand 🡨 kewuk[realPosition]  kewuk[realPosition] 🡨 0  lubang++  **while** hand > 0  **if** lubang **MOD** 8 = 0  **if** turn = 1 **AND** lubang = 8  P[0].bijiCollected 🡨 P[0].bijiCollected + 1  **else** **if** lubang = 16 **AND** turn = 0  P[1].bijiCollected 🡨 P[1].bijiCollected + 1  **else**  hand++  **endif**  **else** **if** lubang > 9  realPosition++  kewuk[realPosition] 🡨 kewuk[realPosition]+1  **else** **if** lubang = 9  realPosition 🡨 0  kewuk[realPosition] 🡨 kewuk[realPosition]+1  **else** **if** lubang = 1  realPosition🡨13  kewuk[realPosition] 🡨 kewuk[realPosition]+1  **else**  realPosition--  kewuk[realPosition] 🡨 kewuk[realPosition]+1  **endif**  hand--  **if** lubang = 16  lubang 🡨 1  **else**  lubang++  **endif**  board(lubang-1)  Sleep(900)  **endwhile**    **if** lubang-1=0  gotoxy(43,15)  **write(layar)** P[1].bijiCollected  gotoxy(45,16)  out🡨false  **else** **if** lubang-1=8  gotoxy(43,15)  **write(layar)** Posisi Terakhir biji  gotoxy(45,16)  out🡨false  **else**  **if** kewuk[realPosition] = 1  out🡨false  turn 🡨 (turn+1) **MOD** 2  **endif**  **endif**  lubang 🡨 lubang - 1  **while** out  **if** kewuk[realPosition] =1 **AND** lubang != 0 **AND** lubang != 8  tembak(lubang,player)  **if** player = 1  P[0].numberOfShoot++  **endif**  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | pickKosong |
| Deskripsi | Memeriksa apa lubang yang dipilih kosong |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Fungsi |
| *input* | **Pick**: integer, variabel penampung lubang yang dipilih pemain |
| *Output* | boolean yang menyatakan kosong atau tidak kuwuk |
| Kamus Data | |
| realPosition: **integer** | |
| Algoritma | |
| **if** pick > 8  realPosition🡨pick-9  **else**  realPosition🡨14-pick  **endif**  **If** kewuk[realPosition] = 0  **return** true  **else**  **return** false  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | Tembak |
| Deskripsi | Memeriksa apa lubang yang dipilih kosong |
| *Author* | Anggita Intan F, Ilham Gibran A.M |
| Jenis | Prosedur |
| *Parameter input* | **Giliran:** integer, penampung nilai yang menyatakan giliran pemain |
| *Parameter Input/Output* | **Lubang:** jumlah biji pada lubang di papan congklak |
| Kamus Data | |
| realPosition: **integer**  tembak: **integer**  lumbung: **integer** | |
| Algoritma | |
| lumbung 🡨 2  **if** lubang > 8  realPosition 🡨 lubang - 9  **else**  realPosition 🡨 14-lubang  **endif**  **if** giliran = 1 **AND** lubang < 8  tembak 🡨 realPosition - 7  **if** kewuk[tembak] > 0  lumbung 🡨 0  **endif**  **else** **if** giliran = 0 **AND** lubang > 8  tembak 🡨 realPosition + 7  **if** kewuk[tembak] > 0  lumbung 🡨 1  **endif**  **else**  gotoxy(35,15)  **write(layar)** pemberitahuan tidak bisa menembak  gotoxy(45,16)  system("pause")  **endif**  **if** lumbung < 2  P[lumbung].bijiCollected 🡨 P[lumbung].bijiCollected +  kewuk[realPosition] + kewuk[tembak]  kewuk[realPosition] 🡨 0  kewuk[tembak] 🡨 0  board(-1)  gotoxy(30,15)  **write(layar)** informasi hasil menembak  gotoxy(45,18)  system("pause")  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | openScore |
| Deskripsi | Membuka data highscore |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Fungsi |
| *Initial State* | Variabel global penampung highscore kosong |
| *Final State* | Variabel global penampung highscore berisi data highscore |
| *output* | **Nomor:**  integer, penampung panjang data highscore |
| Algoritma | |
| **While** read(file highscore)  nomor++  **endwhile**  **return** nomor | |

|  |  |
| --- | --- |
| Nama Modul | saveScore |
| Deskripsi | Menulis data highscore |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Prosedur |
| *Initial State* | Data highscore belum ditulis di file *highscore* |
| *Final State* | Data highscore ditulis di file *highscore* |
| Algoritma | |
| raw.name 🡨 P[0].nama  raw.bijiCollected 🡨 P[0].bijiCollected  raw.numberOfShoot 🡨 P[0].numberOfShoot  raw.numberOfTake 🡨 P[0].numberOfTake  raw.status 🡨 P[0].status  **write(file highscore)** raw.bijiCollected, raw.numberOfShoot,  raw.numberOfTake, raw.status | |

|  |  |
| --- | --- |
| Nama Modul | Swap |
| Deskripsi | Menulis data highscore |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Prosedur |
| *Initial State* | Isi variabel belum ditukar |
| *Final State* | Isi variabel ditukar |
| Algoritma | |
| Temp 🡨 show[i]  show[i] 🡨 show[j]  show[j] 🡨 temp | |

|  |  |
| --- | --- |
| Nama Modul | cekSwap |
| Deskripsi | Menghasilkan boolean untuk menentukan apa terjadinya operasi *swap* berdasarkan input *user* |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Prosedur |
| *Input* | **I:** integer, nomor ruang array 1  **J**: integer**,** nomor ruang array 2  **Choice:**  integer, mewakili pilihan sort berdasarkan apa |
| *Output* | Boolean yang menentukan dilakukan atau tidak proses *swap* nilai |
| Algoritma | |
| **If** choice = 1  **return** strcmp((show[j].name),(show[i].name))<0  **else** **if** choice = 2  **return** show[j].bijiCollected>show[i].bijiCollected  **else** **if** choice = 3  **return** show[j].numberOfShoot>show[i].numberOfShoot  **else** **if** choice = 4  **return** show[j].numberOfTake<show[i].numberOfTake  **endif** | |

|  |  |
| --- | --- |
| Nama Modul | sortHighscore |
| Deskripsi | Mengatur proses sorting *highscore* |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Prosedur |
| *Input* | **I:** integer, nomor ruang array 1  **J**: integer**,** nomor ruang array 2 |
| *Output* | *Highscore* ter-*sort* sesuai pilihan |
| Algoritma | |
| **For** i 🡨 0 to N-2  **for** j 🡨 i+1 to N-1  **if** cekSwap(i,j,choice)  swap(i,j)  **endif**  **endfor**  **endfor** | |

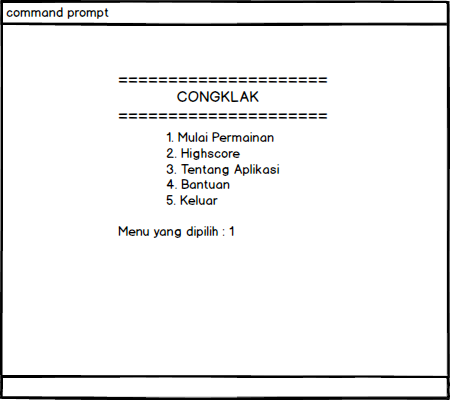
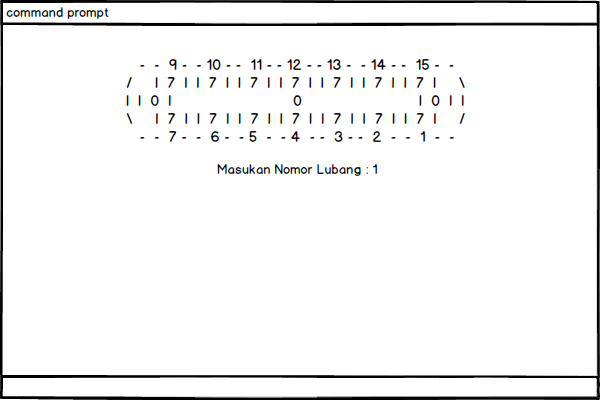
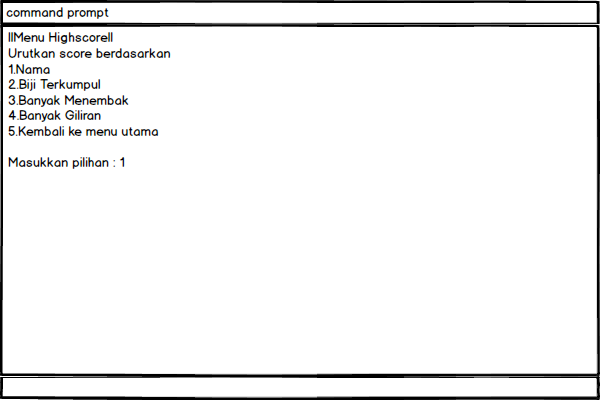
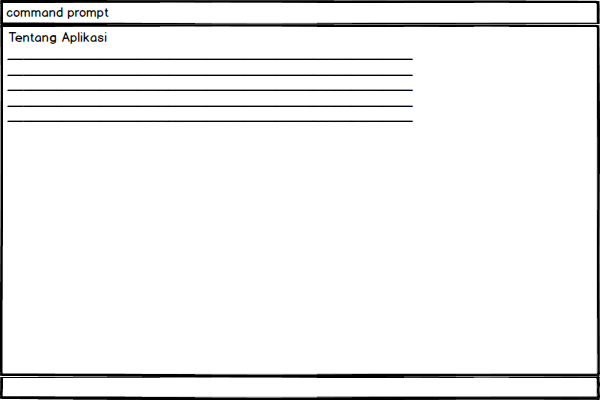
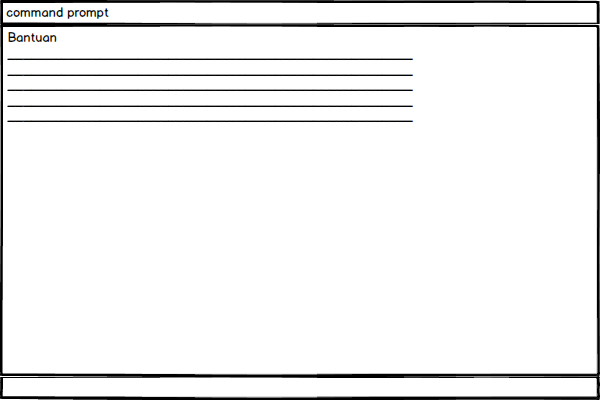
|  |  |
| --- | --- |
| Nama Modul | ShowHighscore |
| Deskripsi | Modul utama proses penampil *highscore* |
| *Author* | Hamzah Prasetio Utomo |
| Jenis | Prosedur |
| *Initial State* | Tampilan menu utama |
| *Final State* | Tampilan *highscore* permainan |
| Kamus Data | |
| N, pilihan, i, y : **integer** | |
| Algoritma | |
| Y 🡨 14  N 🡨 openScore()  **do**  **do**  **clearscreen**  gotoxy(48,1)  **write(layar)** menu untuk sorting highscore  gotoxy(48,8)  **input(pilihan)** pilihan *user* untuk sorting highscore  **while** pilihan<1 **OR** pilihan>5  **if** pilihan!=5  sortHighscore(pilihan,N)  **write(layar)** header tabel highscore  **for** i 🡨 0 to N-1  **write(layar)** show[i].name,show[i].bijiCollected  ,show[i].numberOfShoot,show[i].numberOfTake  **if** show[i].status = 1  **write(layar) “**menang”  **else** **if** show[i].status = 2  **write(layar) “**draw“  **else**  **write(layar) “**kalah”  **endif**  y++  **endfor**  **endif**  gotoxy(48,y)  system("pause")  **while** pilihan!=5 | |

|  |  |
| --- | --- |
| Nama Modul | Tentang |
| Deskripsi | Modul untuk proses *about* |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Prosedur |
| *Initial State* | Tampilan menu utama |
| *Final State* | Tampilan *about* |
| Kamus Data | |
| C, tmp : **integer** | |
| Algoritma | |
| **While** (c 🡨 getc(file about)!=EOF) **AND** tmp = 0  **If** c=';'  tmp 🡨 1  **else**  putchar(c)  **endif**  **endwhile** | |

|  |  |
| --- | --- |
| Nama Modul | Bantuan |
| Deskripsi | Modul untuk proses *help* |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Prosedur |
| *Initial State* | Tampilan menu utama |
| *Final State* | Tampilan *help* |
| Kamus Data | |
| C, tmp : **integer** | |
| Algoritma | |
| **While** (c 🡨 getc(filr)) != EOF  **If** c=';'  tmp 🡨 1  **endif**  **if** tmp = 1  putchar(c)  **endif**  **endwhile** | |

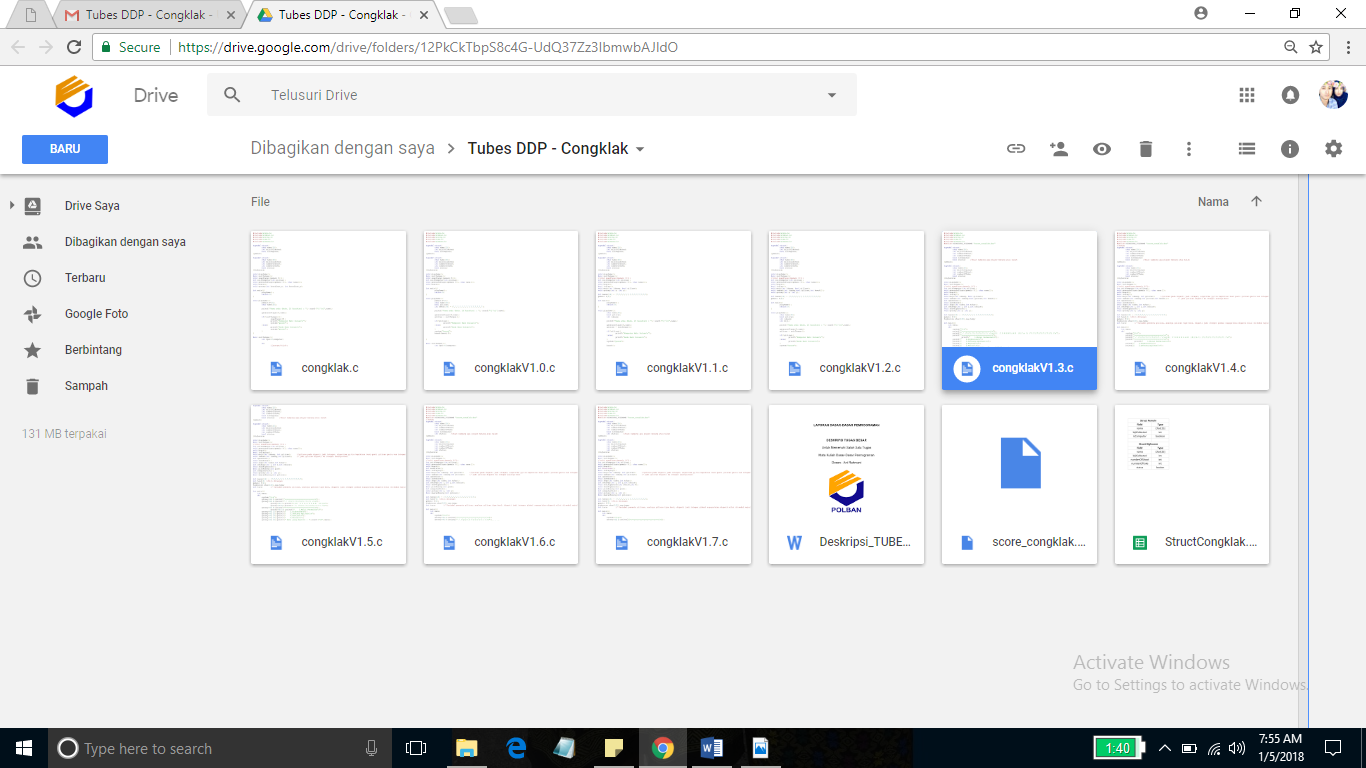
|  |  |
| --- | --- |
| Nama Modul | Gotoxy |
| Deskripsi | Memindahkan lokasi cursor ke koordinat x,y |
| *Author* | Ilham Gibran Achmad Mudzakir |
| Jenis | Prosedur |
| *Initial State* | Cursor terletak pada lokasi default |
| *Final State* | Cursor terletak pada koordinat x,y |
| Kamus Data | |
| X,y : **integer,** koordinat cursor | |
| Algoritma | |
| COORD coord  coord.X 🡨 x  coord.Y 🡨 y  **call** SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord) | |

# BAB V RANCANGAN TAMPILAN

* 1. **Tampilan Utama**
  2. **Tampilan Permainan**
  3. **Tampilan *Highscore***
  4. **Tampilan Tentang Aplikasi**
  5. **Tampilan Bantuan**

# BAB VI PENUTUP

## Alur Kerja dan Pembagian Tugas

Kami melakukan pengerjaan aplikasi congklak ini secara *remote* (jarak jauh) karena terkendala oleh liburan. *Source code* disimpan di Google Drive agar setiap anggota kelompok dapat mengaksesnya. Setiap anggota yang hendak mengedit atau melakukan perubahan terhadap *source code* harus berkomunikasi dengan anggota kelompok yang lain agar tidak terjadi pengerjaan dan perubahan pada *file* di saat yang bersamaan, setiap anggota juga diwajibkan melakukan *versioning* saat melakukan perubahan pada *source code.*

Kelompok kami membagi pengerjaan tugas sebagai berikut. Anggita melakukan perancangan untuk sistem bermain congklak serta melakukan perancangan pada skenario program, membuat modul tembak() dan melakukan dokumentasi pengerjaan aplikasi serta melakukan *testing*. Hamzah melakukan pengerjaan pada modul yang berhubungan dengan *Highscore* dan akses ke *file*, lalu melakukan penyelesaian pada modul playgame(). Ilham melakukan perancangan playgame() lalu membuat *structure chart,* melakukan pengerjaan pada proses suit dan sebagian dari playgame(), membuat about dan bantuan mengenai cara bermain, dan membuat mockup tampilan.

## Lesson Learned

1. Anggita Intan Fania

Menjadi lebih belajar tentang modular, dapat belajar bagaimana cara membagi tugas dan menggabungkan modul, belajar dalam membagi waktu ketika liburan, memahami pentingnya *tracing* dandokumentasi, mengenal fungsi gotoxy dan belajar bagaimana membuat board, mempelajari *source code* orang lain dan belajar tentang *file* pada penyimpanan *highscore.*

1. Hamzah Prasetio Utomo
2. Belajar memanajemen waktu
3. Belajar kerja sama pembuatan program dalam sebuah tim
4. Mengerti pentingnya perencanaan struktur program sebelum mulai koding
5. Memahami kapan diperlukannya variabel global dan lokal
6. Ilham Gibran Achmad Mudzakir

Melalui tugas besar ini saya mempelajari cara memanajemen pengerjaan suatu aplikasi secara jarak jauh (*remote*),serta mengenal dan melakukan perancangan pembuatan aplikasi secara berkelompok, dari mulai merancang skenario dari aplikasi yang akan dibuat, serta mengeksekusinya, dan saya merasa cukup terbantu dengan adanya perancangan tersebut saat melakukan implementasi ke dalam bahasa pemrograman. Dalam hal teknis saya mempelajari beberapa sintaks baru seperti Sleep(), lalu saya juga mempelajari mengenai HANDLE, untuk mengubah warna dari teks pada *command prompt*.