

# **LAPORAN TUGAS BESAR**

## **IF2111 Algoritma dan Struktur Data**


### **Console 'BNMO'**

Dipersiapkan oleh:

Kelompok 07

18221067	Fawwaz Abrial Saffa
18221069	Gibran Fasha Ghazanfar
18221087	Willy Frans Farel Sijabat
18221101	Ilmagita Nariswari
18221163	Aufar Ramadhan

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung  
Jl. Ganesha 10, Bandung 40132

	<b>Sekolah Teknik Elektro dan Informatika ITB</b>	<b>Nomor Dokumen</b>		<b>Halaman</b>
		<i>IF2111-TB1-07</i>		39
		<i>Revisi</i>	<i>&lt;no revisi&gt;</i>	11 November 2022

# Daftar Isi

<b>1 Ringkasan</b>	<b>3</b>
<b>2 Penjelasan Tambahan Spesifikasi Tugas</b>	<b>4</b>
2.1 Bonus: Game 2048	4
<b>3 Struktur Data (ADT)</b>	<b>5</b>
3.1 ADT Array	5
3.1.1 Sketsa Struktur Data	5
3.1.2 Persoalan yang Diselesaikan	7
3.1.3 Alasan Pemilihan	7
3.1.4 Implementasi	7
3.2 ADT Matriks	7
3.2.1 Sketsa Struktur Data	7
3.2.2 Persoalan yang Diselesaikan	9
3.2.3 Alasan Pemilihan	9
3.2.4 Implementasi	10
3.3 ADT mesinkarakter	10
3.3.1 Sketsa Struktur Data	10
3.3.2 Persoalan yang Diselesaikan	11
3.3.3 Alasan Pemilihan	11
3.3.4 Implementasi	11
3.4 ADT mesinkata	11
3.4.1 Sketsa Struktur Data	11
3.4.2 Persoalan yang Diselesaikan	13
3.4.3 Alasan Pemilihan	13
3.4.4 Implementasi	13
3.5 ADT Queue	13
3.5.1 Sketsa Struktur Data	13
3.5.2 Persoalan yang Diselesaikan	17
3.5.3 Alasan Pemilihan	18
3.5.4 Implementasi	18
<b>4 Program Utama</b>	<b>18</b>
4.1 Main Menu	18
4.2 Pembacaan dan Inisialisasi File Konfigurasi	18
4.3 Pemanggilan Command	19

4.4 Game RNG	20
4.5 Game Diner Dash	20
4.6 Game 2048	20
<b>5 Algoritma-Algoritma Menarik</b>	<b>21</b>
5.1 Force Delete Queue	21
5.2 Algoritma scanf() di C	21
<b>6 Data Test</b>	<b>21</b>
<b>Fitur yang diteset berupa perintah command yang akan dipakai ketika program berjalan.</b>	<b>21</b>
6.1 START	21
6.2 LOAD	22
6.3 SAVE	22
6.4 CREATE GAME	23
6.5 LIST GAME	23
6.6 DELETE GAME	24
6.7 QUEUE GAME	25
6.8 PLAY GAME	26
6.9 SKIP GAME	27
6.10 QUIT	28
6.11 HELP	29
<b>7 Test Script</b>	<b>29</b>
<b>8 Pembagian Kerja dalam Kelompok</b>	<b>33</b>
<b>9 Lampiran</b>	<b>34</b>
9.1 Deskripsi Tugas Besar 1	34
9.2 Notulen Rapat	34
9.3 Log Activity Anggota Kelompok	35
9.4 Asistensi	36

# 1 Ringkasan

Indra dan Doni memiliki sebuah robot video game console bernama BNMO (dibaca: Binomo). Dua bulan lalu, BNMO mengalami kerusakan dan berhasil diperbaiki. Akan tetapi, setelah perbaikan, BNMO mendapatkan lebih banyak *bug* dalam sistemnya. Oleh karena itu, Indra dan Doni sedang mencari programmer yang lebih handal untuk memprogram ulang robot video game console kesayangan mereka.

BNMO adalah sebuah video game console berbasis CLI (Command Line Interface) yang diimplementasikan menggunakan bahasa C. Fitur-fitur utama BNMO adalah memainkan game, menambahkan game, menghapus game, dan mengurutkan game-game yang akan dimainkan. Saat pertama kali dinyalakan, user akan melihat tampilan Welcome Page dan beberapa tampilan menu seperti START dan LOAD. Setelah itu, BNMO akan menerima input-input commands lainnya.

Laporan ini berisi penjelasan tentang game yang telah kami buat. Bagian pertama berisi ringkasan berisi deskripsi umum persoalan dan isinya secara singkat. Bagian kedua berisi tentang penjelasan tambahan spesifikasi tugas. Bagian ketiga menceritakan struktur data yang dipakai pada program. Bagian keempat berisikan algoritma-algoritma menarik yang ditemukan pada program. Bagian kelima berisi tentang data test. Bagian keenam berisi tentang test script. Bagian ketujuh berisikan tentang test script. Bagian kedelapan berisikan tentang pembagian kerja.

Permainan ini digunakan menggunakan bahasa C dengan struktur data yang telah dipelajari pada mata kuliah IF2111 Algoritma dan Struktur Data. ADT-ADT yang digunakan yaitu array, matriks, mesin karakter dan mesin kata, dan queue.

## 2 Penjelasan Tambahan Spesifikasi Tugas

### 2.1 Bonus: Game 2048

Game 2048 adalah game *sliding tile*. Game dimulai dalam sebuah grid 4x4, dan sebuah angka berkelipatan 2 akan digenerasikan secara random pada salah satu kotak grid. User kemudian bisa mengetik W (atas), A (kiri), S (bawah), dan D (kanan) untuk menggeser angka-angka pada grid ke atas, kiri, bawah, atau kanan. Apabila ada angka yang bertabrakan dengan angka lainnya dan kedua angka tersebut bernilai sama, maka angka tersebut akan bergabung dan nilai yang baru merupakan jumlah kedua angka tersebut di tempat yang baru. Selain itu juga terdapat fitur UNDO yang memungkinkan pemain untuk kembali ke state sebelumnya yang hanya dapat digunakan maksimal sekali dalam setiap *turn*.

## 3 Struktur Data (ADT)

### 3.1 ADT Array

#### 3.1.1 Sketsa Struktur Data

```
#include "boolean.h"
#include "mesinkata.h"
#include "../functions.h"

#ifndef ARRAY_H
#define ARRAY_H

/* Kamus Umum */

#define IdxMax 100
#define IdxMin 1
#define IdxUndef -999 /* indeks tak terdefinisi*/

/* Definisi elemen dan koleksi objek */
typedef int IdxTypeArray;
typedef Word ElTypeArray;

typedef struct
{
    ElTypeArray TI [IdxMax-IdxMin+1]; /* memori tempat menyimpan elemen
(container) */
    int Neff; /* banyaknya elemen efektif */
} TabWord;

/* Indeks yang digunakan [IdxMin..IdxMax] */
/* Jika T adalah TabInt, cara deklarasi dan akses: */
/* Deklarasi : T : TabInt */
/* Maka cara akses:
 * T.Neff untuk mengetahui banyaknya elemen
 * T.TI untuk mengakses seluruh nilai elemen tabel
 * T.TI[i] untuk mengakses elemen ke-i */
/* Definisi :
 * Tabel kosong: T.Neff = 0
 * Definisi elemen pertama : T.TI[i] dengan i=1
 * Definisi elemen terakhir yang terdefinisi: T.TI[i] dengan i=T.Neff */

/* ***** KONSTRUKTOR ***** */
/* Konstruktor : create tabel kosong */
void MakeEmpty (TabWord *T);
/* I.S. sembarang */
/* F.S. Terbentuk tabel T kosong dengan kapasitas IdxMax-IdxMin+1 */

/* ***** SELEKTOR ***** */
```

```

/* *** Banyaknya elemen *** */
int NbElmt (TabWord T);
/* Mengirimkan banyaknya elemen efektif tabel */
/* Mengirimkan nol jika tabel kosong */
/* *** Daya tampung container *** */
int MaxNbEl (TabWord T);
/* Mengirimkan maksimum elemen yang dapat ditampung oleh tabel */
/* *** Selektor INDEKS *** */
IdxTypeArray GetFirstIdx (TabWord T);
/* Prekondisi : Tabel T tidak kosong */
/* Mengirimkan indeks elemen pertama */
IdxTypeArray GetLastIdx (TabWord T);
/* Prekondisi : Tabel T tidak kosong */
/* Mengirimkan indeks elemen terakhir */
/* *** Menghasilkan sebuah elemen *** */
ElTypeArray GetElmt (TabWord T, IdxTypeArray i);
/* Prekondisi : Tabel tidak kosong, i antara FirstIdx(T)..LastIdx(T) */
/* Mengirimkan elemen tabel yang ke-i */
int GetElmtIdx (TabWord T, Word elmt);
/* Prekondisi: bebas
   Mengirimkan indeks elemen tabel pertama yang berisi sama dengan elmt.
   Mengirimkan IdxUndef (-999) jika tidak ditemukan (hilang coy) */

/* *** Selektor SET : Mengubah nilai TABEL dan elemen tabel *** */
/* Untuk type private/limited private pada bahasa tertentu */
void SetTab (TabWord Tin, TabWord *Tout);
/* I.S. Tin terdefinisi, sembarang */
/* F.S. Tout berisi salinan Tin */
/* Assignment THsl -> Tin */
void SetEl (TabWord *T, IdxTypeArray i, ElTypeArray v);
/* I.S. T terdefinisi, sembarang */
/* F.S. Elemen T yang ke-i bernilai v */
/* Mengeset nilai elemen tabel yang ke-i sehingga bernilai v */
void SetNeff (TabWord *T, IdxTypeArray N);
/* I.S. T terdefinisi, sembarang */
/* F.S. Nilai indeks efektif T bernilai N */
/* Mengeset nilai indeks elemen efektif sehingga bernilai N */

/* ***** Test Indeks yang valid ***** */
boolean IsIdxValid (TabWord T, IdxTypeArray i);
/* Prekondisi : i sembarang */
/* Mengirimkan true jika i adalah indeks yang valid utk ukuran tabel */
/* yaitu antara indeks yang terdefinisi utk container*/
boolean IsIdxEff (TabWord T, IdxTypeArray i);
/* Prekondisi : i sembarang */
/* Mengirimkan true jika i adalah indeks yang terdefinisi utk tabel */
/* yaitu antara FirstIdx(T)..LastIdx(T) */

/* ***** TEST KOSONG/PENUH ***** */
/* *** Test tabel kosong *** */
boolean IsEmpty (TabWord T);

```

```

/* Mengirimkan true jika tabel T kosong, mengirimkan false jika tidak */
/* *** Test tabel penuh *** */
boolean IsFull (TabWord T);
/* Mengirimkan true jika tabel T penuh, mengirimkan false jika tidak */

/* ***** BACA dan TULIS dengan INPUT/OUTPUT device ***** */
void DisplayArray (TabWord T);
/* Proses : Menuliskan isi tabel dengan traversal */
/* I.S. T boleh kosong */
/* F.S. Jika T tidak kosong : indeks dan elemen tabel ditulis berderet ke
bawah */
/* Jika isi tabel [1,2,3] maka akan diprint
0:1
1:2
2:3
*/
/* Jika T kosong : Hanya menulis "Tabel kosong" */

/* INSERT & DELETE */
void InsertLast (TabWord *T, ElTypeArray X); // Insert di akhir array

void DeleteAt (TabWord *T, int id); // Hapus elemen array T pada indeks
ke-id. Array tidak kosong

```

### 3.1.2 Persoalan yang Diselesaikan

ADT array digunakan untuk menampilkan daftar game yang ada di dalam sistem.

### 3.1.3 Alasan Pemilihan

Dipilih ADT array statik dan bukan dinamis karena penggunaan array dinamis dirasa terlalu berlebihan jika dipakai hanya untuk menyimpan daftar game. Sebagai pengganti, kami membuat ADT array statik dengan maksimum 150 elemen, jika pengguna ingin menambahkan game ketika array sudah penuh, pengguna harus menghapus game terlebih dahulu.

### 3.1.4 Implementasi

Implementasi array dalam program ini ada di dalam kode array.c dalam folder ADT (/src/ADT/array.c)

## 3.2 ADT Matriks

### 3.2.1 Sketsa Struktur Data

```

/* ***** Definisi TYPE MATRIKS dengan indeks dan elemen integer
***** */

```

```

/* Ukuran minimum dan maksimum baris dan kolom */
#define MaxElBrs 4
#define MaxElKol 4

typedef int indeksMatriks; /* indeks baris, kolom */
typedef int ElTypeMatriks;
typedef struct {
    ElTypeMatriks Mem[MaxElBrs][MaxElKol];
    int NBrEff; /* banyaknya/ukuran baris yg terdefinisi */
    int NKolEff; /* banyaknya/ukuran kolom yg terdefinisi */
} Matriks;

#define NBrEff(M) (M).NBrEff
#define NKolEff(M) (M).NKolEff
#define Elmt(M,i,j) (M).Mem[(i)][(j)]

/* ADT Matriks ini dibuat untuk kepentingan game 2048, sehingga banyak
primitif yang tidak diimplementasikan karena tidak dibutuhkan (terutama
fungsi matematis seperti transpose atau determinan) dan banyak juga primitif
yang diimplementasikan dengan membayangkan game 2048 sebagai pengguna ADT
utama. */

/***** PRIMITIF ASLI MATRIKS *****/
void MakeMatriks (int NB, int NK, Matriks * M);
/* Membentuk sebuah MATRIKS "kosong" yang siap diisi berukuran NB x NK di
"ujung kiri" memori
I.S. : NB dan NK adalah valid untuk memori matriks yang dibuat
F.S. : Matriks M sesuai dengan definisi di atas terbentuk */

boolean isMatriksEmpty(Matriks M);
/* Mengembalikan true apabila matriks kosong.
Syarat dari matriks kosong adalah seluruh elemen dalam matriks bernilai
0. */

boolean isMatriksFull(Matriks M);
/* Mengembalikan true apabila matriks sudah penuh.
Syarat dari matriks kosong adalah tidak ada elemen dalam matriks bernilai
0. */

int maxInMatriks(Matriks M);
/* Mengembalikan nilai terbesar dalam matriks. */

void copyMatriks(Matriks in, Matriks* out);
/* Menyalin seluruh isi Matriks in ke Matriks out.
I.S. : in dan out sembarang
F.S. : in dan out memiliki isi yang sama */

void displayMatriks(Matriks M);
/* Mencetak Matriks ke layar dalam bentuk kotak sebagai tampilan game 2048.
I.S. : Sembarang

```



```

F.S. : Seluruh isi matriks tercetak ke layar */

/***** PRIMITIF MATRIKS UNTUK 2048 *****/
void geserMatriksKanan(Matriks* M, int* score, boolean merge);
/* Menggeser seluruh isi matriks ke kanan dengan bekas tempat elemen
pergeseran menjadi 0.
I.F. : Sembarang
F.S. : Seluruh elemen akan tergeser ke pojok kanan matriks, bila
mergbernilai true maka jika dalam penggeseran elemen sebelah kanannya
memiliki nilai yang sama, nilai barunya adalah penjumlahan nilai sekarang
dengan nilai kanannya. */

void geserMatriksKiri(Matriks* M, int* score, boolean merge);
/* Menggeser seluruh isi matriks ke kiri dengan bekas tempat elemen
pergeseran menjadi 0.
I.F. : Sembarang
F.S. : Seluruh elemen akan tergeser ke pojok kiri matriks, bila merge
bernilai true maka jika dalam penggeseran elemen sebelah kirinya memiliki
nilai yang sama, nilai barunya adalah penjumlahan nilai sekarang dengan
nilai kirinya. */

void geserMatriksAtas(Matriks* M, int* score, boolean merge);
/* Menggeser seluruh isi matriks ke atas dengan bekas tempat elemen
pergeseran menjadi 0.
I.F. : Sembarang
F.S. : Seluruh elemen akan tergeser ke pojok atas matriks, bila merge
bernilai true maka jika dalam penggeseran elemen sebelah atasnya memiliki
nilai yang sama, nilai barunya adalah penjumlahan nilai sekarang dengan
nilai atasnya. */

void geserMatriksBawah(Matriks* M, int* score, boolean merge);
/* Menggeser seluruh isi matriks ke bawah dengan bekas tempat elemen
pergeseran menjadi 0.
I.F. : Sembarang
F.S. : Seluruh elemen akan tergeser ke pojok bawah matriks, bila merge
bernilai true maka jika dalam penggeseran elemen sebelah bawahnya memiliki
nilai yang sama, nilai barunya adalah penjumlahan nilai sekarang dengan
nilai bawahnya. */

```

### 3.2.2 Persoalan yang Diselesaikan

ADT Matriks ini dibuat untuk kepentingan game 2048, sehingga banyak primitif yang tidak diimplementasikan karena tidak dibutuhkan (terutama fungsi matematis seperti transpose atau determinan) dan banyak juga primitif yang diimplementasikan dengan membayangkan permainan 2048 sebagai pengguna utama ADT.

### 3.2.3 Alasan Pemilihan

Dalam permainan 2048 *board*-nya berbentuk grid seperti sebuah matriks. Salah satu aturan permainan adalah pemain dapat menggeser seluruh elemen yang berada di dalam matriks ke salah satu sisi (atas, bawah, kiri, atau kanan) dan jika ada dua elemen yang sama maka akan digabungkan.

Dipilih ADT matriks karena board dari permainan sudah berbentuk matriks dan aturan-aturan permainan seperti penggeseran dan penggabungan elemen dapat diimplementasi dengan mudah.

### 3.2.4 Implementasi

Implementasi array terdapat dalam kode matriks.c pada 'src/ADT/matriks.c'.

## 3.3 ADT mesinkarakter

### 3.3.1 Sketsa Struktur Data

```
/* File: mesinkarakter.h */
/* Definisi Mesin Karakter */

#ifndef __MESIN_KAR_H_
#define __MESIN_KAR_H_

#include "boolean.h"
#include "mesinkata.h"

#define MARK '\n'
/* State Mesin */
extern char currentChar;
extern boolean EOP;

void RESETPITA();
/* Mengembalikan state awal pita (kosong tanpa inputan) supaya pita dapat
   digunakan kembali tanpa overflow dari pita input sebelumnya.
   I.S. : sembarang
   F.S. : pita kosong (NULL) */

void STARTFILE(char* file_name);
/* Mesin siap dioperasikan. Pita disiapkan untuk dibaca.
   Pita yang akan diambil adalah pita yang berada di dalam file
   pada path ../data/file_name.
   Pita baca diambil dari fopen(path).
   I.S. : sembarang
   F.S. : currentChar adalah karakter pertama pada pita
        Jika currentChar != MARK maka EOP akan padam (false)
        Jika currentChar = MARK maka EOP akan menyala (true) */

void START();
/* Mesin siap dioperasikan. Pita disiapkan untuk dibaca.
```

```

    Karakter pertama yang ada pada pita posisinya adalah pada jendela.
    Pita baca diambil dari stdin. Sebelum pembacaan, pita akan
    dikembalikan ke state awal.
    I.S. : sembarang
    F.S. : currentChar adalah karakter pertama pada pita
           Jika currentChar != MARK maka EOP akan padam (false)
           Jika currentChar = MARK maka EOP akan menyala (true) */

void ADV();
/* Pita dimajukan satu karakter.
   I.S. : Karakter pada jendela = currentChar, currentChar != MARK
   F.S. : currentChar adalah karakter berikutnya dari currentChar yang lama,
          currentChar mungkin = MARK
          Jika currentChar = MARK maka EOP akan menyala (true) */

#endif

```

### 3.3.2 Persoalan yang Diselesaikan

ADT mesin karakter digunakan sebagai pengganti dari scanf dan juga sebagai persyaratan dari ADT mesin kata.

### 3.3.3 Alasan Pemilihan

ADT mesinkarakter digunakan untuk membaca input dan command dari user.

### 3.3.4 Implementasi

ADT mesinkarakter diimplementasikan dalam 'src/ADT/mesinkarakter.c'.

## 3.4 ADT mesinkata

### 3.4.1 Sketsa Struktur Data

```

/* File: mesinkata.h */
/* Definisi Mesin Kata: Model Akuisisi Versi I */

#ifndef __MESINKATA_H__
#define __MESINKATA_H__

#include "boolean.h"
#include "mesinkarakter.h"

#define NMax 150
#define BLANK ' '

typedef struct
{
    char TabWord[NMax]; /* container penyimpan kata, indeks yang dipakai

```

```

[0..NMax-1] */
    int Length;
} Word;

/* State Mesin Kata */
extern boolean EndWord;
extern Word currentWord;

void ResetCurrentWord();
/* Mengembalikan currentWord ke state awal (kosong / length=0).
    I.S. : currentWord sembarang
    F.S. : currentWord kosong (memiliki length 0) */

void IgnoreBlanks();
/* Mengabaikan satu atau beberapa BLANK.
    I.S. : currentChar sembarang
    F.S. : currentChar ≠ BLANK atau currentChar = MARK */

void STARTWORD();
/* Kata dibaca dengan prosedur START() yang akan membaca dari input
    user, akuisisi kata menggunakan CopyWord.
    I.S. : currentChar sembarang
    F.S. : EndWord = true, dan currentChar = MARK;
           atau EndWord = false, currentWord adalah kata yang sudah diakuisisi,
           currentChar karakter pertama sesudah karakter terakhir kata */

void ADVWORD();
/* I.S. : currentChar adalah karakter pertama kata yang akan diakuisisi
    F.S. : currentWord adalah kata terakhir yang sudah diakuisisi,
           currentChar adalah karakter pertama dari kata berikutnya, mungkin MARK
           Jika currentChar = MARK, EndWord = true.
    Proses : Akuisisi kata menggunakan procedure SalinWord */

void CopyWord();
/* Mengakuisisi kata, menyimpan dalam currentWord.
    I.S. : currentChar adalah karakter pertama dari kata
    F.S. : currentWord berisi kata yang sudah diakuisisi;
           currentChar = BLANK atau currentChar = MARK;
           currentChar adalah karakter sesudah karakter terakhir yang diakuisisi.
           Jika panjang kata melebihi NMax, maka sisa kata "dipotong" */

void STARTWORDFILE(char* path);
/* Kata dibaca dengan prosedur STARTFILE() yang akan membaca dari file pada
    path,
    akuisisi kata menggunakan CopyWordWithBlanks.
    I.S. : currentChar sembarang
    F.S. : EndWord = true, dan currentChar = MARK;
           atau EndWord = false, currentWord adalah kata yang sudah diakuisisi,
           currentChar karakter pertama sesudah karakter terakhir kata */

void ADVWORDFILE();

```

```

/* I.S. : currentChar adalah karakter pertama kata yang akan diakuisisi
   F.S. : currentWord adalah kata terakhir yang sudah diakuisisi,
         currentChar adalah karakter pertama dari kata berikutnya, mungkin MARK
   Jika currentChar = MARK, EndWord = true.
   Proses : Akuisisi kata menggunakan procedure CopyWordWithBlanks */

void CopyWordWithBlanks();
/* Mengakuisisi kata dengan membolehkan blanks, menyimpan dalam currentWord.
   I.S. : currentChar adalah karakter pertama dari kata
   F.S. : currentWord berisi kata yang sudah diakuisisi;
         currentChar = BLANK atau currentChar = MARK;
         currentChar adalah karakter sesudah karakter terakhir yang diakuisisi.
         Jika panjang kata melebihi NMax, maka sisa kata "dipotong" */

void STARTLINE();
/* Membaca satu line dari user, dengan membolehkan spasi.
   I.S. : currentChar sembarang
   F.S. : EndWord = true, dan currentChar = MARK;
         atau EndWord = false, currentWord adalah kata yang sudah diakuisisi,
         currentChar karakter pertama sesudah karakter terakhir kata */

boolean isKataEqual(Word W1, Word W2);
/* Mengembalikan true bila Word W1 sama dengan Word W2. */

boolean isKataInt(Word W);
/* Mengembalikan true apabila seluruh isi Word berupa numerik */

void displayWord(Word W);
/* Menampilkan isi Word ke layar.
   I.S. : sembarang
   F.S. : Seluruh isi dalam Word telah ditampilkan pada layar */

#endif

```

### 3.4.2 Persoalan yang Diselesaikan

ADT mesin kata digunakan sebagai pengganti dari scanf, dan hal ini dikembangkan untuk membuat fungsi scan sendiri.

### 3.4.3 Alasan Pemilihan

ADT mesinkata digunakan untuk menerima input dan command dari user.

### 3.4.4 Implementasi

ADT mesinkata diimplementasikan dalam 'src/ADT/mesinkata.c'.

## 3.5 ADT Queue

### 3.5.1 Sketsa Struktur Data

```
/* File : queue.h */
/* Definisi ADT Queue dengan representasi array secara eksplisit dan alokasi
statik */

#ifndef QUEUE_H
#define QUEUE_H

#include "boolean.h"
#include "mesinkata.h"
#include "../functions.h"

#define IDX_UNDEF -1
#define CAPACITY_QUEUE 100

/* Definisi elemen dan address */
typedef Word ElTypeQueue;
typedef struct {
    ElTypeQueue buffer[CAPACITY_QUEUE];
    int idxHead;
    int idxTail;
} Queue;

/* ***** AKSES (Selektor) ***** */
/* Jika q adalah Queue, maka akses elemen : */
#define IDX_HEAD(q) (q).idxHead
#define IDX_TAIL(q) (q).idxTail
#define HEAD(q) (q).buffer[(q).idxHead]
#define TAIL(q) (q).buffer[(q).idxTail]

/* *** Kreator *** */
void CreateQueue(Queue *q);
/* I.S. sembarang */
/* F.S. Sebuah q kosong terbentuk dengan kondisi sbb: */
/* - Index head bernilai IDX_UNDEF */
/* - Index tail bernilai IDX_UNDEF */
/* Proses : Melakukan alokasi, membuat sebuah q kosong */

/* ***** Prototype ***** */
boolean isEmpty(Queue q);
/* Mengirim true jika q kosong: lihat definisi di atas */
boolean isFull(Queue q);
/* Mengirim true jika tabel penampung elemen q sudah penuh */
/* yaitu IDX_TAIL akan selalu di belakang IDX_HEAD dalam buffer melingkar*/

int length(Queue q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika q kosong. */
```

```

boolean isInQueue(Queue q, Word w);
// Mengirimkan apakah Word w terdapat di dalam queue

/* *** Primitif Add/Delete *** */
void enqueue(Queue *q, ElTypeQueue val);
/* Proses: Menambahkan val pada q dengan aturan FIFO */
/* I.S. q mungkin kosong, tabel penampung elemen q TIDAK penuh */
/* F.S. val menjadi TAIL yang baru, IDX_TAIL "mundur" dalam buffer
melingkar. */

void dequeue(Queue *q, ElTypeQueue *val);
/* Proses: Menghapus val pada q dengan aturan FIFO */
/* I.S. q tidak mungkin kosong */
/* F.S. val = nilai elemen HEAD pd I.S., IDX_HEAD "mundur";
q mungkin kosong */

/* *** Display Queue *** */
void DisplayQueue(Queue q);
/* Proses : Menuliskan isi Queue dengan traversal, Queue ditulis di antara
kurung
    siku; antara dua elemen dipisahkan dengan separator "koma", tanpa
    tambahan
    karakter di depan, di tengah, atau di belakang, termasuk spasi dan enter
*/
/* I.S. q boleh kosong */
/* F.S. Jika q tidak kosong: [e1,e2,...,en] */
/* Contoh : jika ada tiga elemen bernilai 1, 20, 30 akan dicetak: [1,20,30]
*/
/* Jika Queue kosong : menulis [] */

#endif

```

```

/* File : q_dinerdash.h */
/* Definisi ADT Queue_DD dengan representasi array secara eksplisit dan
alokasi statik */

#ifndef Q_DINERDASH_H
#define Q_DINERDASH_H

#include "boolean.h"

#define IDX_UNDEF -1
#define CAPACITY_QUEUE_DD 20

/* Definisi elemen dan address */
typedef struct {
    char* makanan;
    int durasi;
    int ketahanan;
}

```

```

        int harga;
    } Food;

typedef Food ElTypeQDD;

typedef struct {
    ElTypeQDD buffer[CAPACITY_QUEUE_DD];
    int idxHeadQDD;
    int idxTailQDD;
} Queue_DD;

/* ***** AKSES (Selektor) ***** */
/* Jika q adalah Queue_DD, maka akses elemen : */
#define IDX_HEAD_DD(q) (q).idxHeadQDD
#define IDX_TAIL_DD(q) (q).idxTailQDD
#define HEAD_DD(q) (q).buffer[(q).idxHeadQDD]
#define TAIL_DD(q) (q).buffer[(q).idxTailQDD]

/* *** Kreator *** */
void CreateQueueDD(Queue_DD *q);
/* I.S. sembarang */
/* F.S. Sebuah q kosong terbentuk dengan kondisi sbb: */
/* - Index head bernilai IDX_UNDEF */
/* - Index tail bernilai IDX_UNDEF */
/* Proses : Melakukan alokasi, membuat sebuah q kosong */

/* ***** Prototype ***** */
boolean isEmptyDD(Queue_DD q);
/* Mengirim true jika q kosong: lihat definisi di atas */
boolean isFullDD(Queue_DD q);
/* Mengirim true jika tabel penampung elemen q sudah penuh */
/* yaitu IDX_TAIL_DD akan selalu di belakang IDX_HEAD_DD dalam buffer melingkar */

int lengthDD(Queue_DD q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika q kosong. */

/* *** Primitif Add/Delete *** */
void enqueueDD(Queue_DD *q, ElTypeQDD val);
/* Proses: Menambahkan val pada q dengan aturan FIFO */
/* I.S. q mungkin kosong, tabel penampung elemen q TIDAK penuh */
/* F.S. val menjadi TAIL_DD yang baru, IDX_TAIL_DD "mundur" dalam buffer melingkar. */

void dequeueDD(Queue_DD *q, ElTypeQDD *val);
/* Proses: Menghapus val pada q dengan aturan FIFO */
/* I.S. q tidak mungkin kosong */
/* F.S. val = nilai elemen HEAD_DD pd I.S., IDX_HEAD_DD "mundur"; q mungkin kosong */

void ForceDeleteAt(Queue_DD *q, int i);

```



```

/* Menghapus elemen array secara paksa (bukan dequeueDD) */
/* I.S. Queue_DD terdefinisi dan tidak kosong */
/* F.S. Elemen Queue_DD ke-i terhapus */

/* *** Display Queue_DD *** */
void displayQueueFood(Queue_DD q);
/* Proses : Menuliskan isi Queue_DD dengan traversal */
/* I.S. q boleh kosong */
/* F.S. Jika q kosong akan dicetak: */
/*
Makanan | Durasi memasak | Ketahanan | Harga
-----
          |                |          |
*/
/* Contoh : jika ada tiga elemen Food akan dicetak: */
/*
Makanan | Durasi memasak | Ketahanan | Harga
-----
M0      | 2              | 3          | 15000
M1      | 3              | 1          | 15000
M2      | 1              | 4          | 15000
*/

void displayQueueRTS(Queue_DD q);
/* Proses : Menuliskan isi Queue_DD dengan traversal */
/* I.S. q boleh kosong */
/* F.S. Jika q tidak kosong dan memiliki 1 elemen */
/*
Makanan | Sisa ketahanan makanan
-----
M2      | 4
*/
/* Jika Queue_DD kosong : akan dicetak */
/*
Makanan | Sisa ketahanan makanan
-----
*/

void displayQueueCooked(Queue_DD q);
/* Proses : Menuliskan isi Queue_DD dengan traversal */
/* I.S. q boleh kosong */
/* F.S. Jika q tidak kosong: [e1,e2,...,en] */
/* Contoh : jika ada dua makanan yang sedang dimasak */
/*
Makanan | Sisa durasi memasak
-----
M0      | 1
M1      | 3
*/
/* Jika Queue_DD kosong */
/*

```

```

Makanan | Sisa durasi memasak
-----
*/
#endif

```

### 3.5.2 Persoalan yang Diselesaikan

Persoalan yang diselesaikan dengan Queue adalah mengantri game dalam sistem dan mengantri makanan dalam permainan Diner Dash. ADT queue digunakan untuk mengantri game dalam sistem. Dalam implementasinya, ADT queue terdapat dalam file queue.c dan q\_dinerdash.c. ADT q\_dinerdash digunakan untuk sistem antrian pada game Diner Dash.

### 3.5.3 Alasan Pemilihan

Queue dirasa paling cocok untuk antrian game di sistem karena memiliki sistem FIFO (*First In First Out*). Karena permainan Diner Dash merupakan permainan *management* antrian, queue dirasa paling cocok untuk hal tersebut.

### 3.5.4 Implementasi

Implementasi ADT queue terdapat dalam 'src/ADT/queue.c' dan 'src/DADT/q\_dinerdash.c'.

## 4 Program Utama

### 4.1 Main Menu

Main menu berperan sebagai fungsi main pada program ini. Program main menu memiliki fungsi untuk menampilkan landing page, menerima input command dari user, melakukan looping program sampai suatu kondisi terpenuhi, dan melakukan pemanggilan pemanggilan fungsi sesuai inputan user. Program main menggunakan ADT mesin kata dan ADT mesin karakter untuk menerima inputan user. terdapat 2 jenis input command yakni input command yang valid dan input command tidak valid. Input command dikatakan valid jika merupakan salah satu dari command yang ada, sedangkan command yang tidak valid adalah inputan yang tidak terdapat didalam command. Inputan command valid akan di eksekusi dengan memanggil fungsi yang sesuai dengan inputan. Pada awal di run program input command yang valid ialah "START", "LOAD", dan "HELP" dan input selain ketiga command akan dianggap sebagai input command tidak valid. Setelah memberikan input "START" atau "LOAD" maka command yang terdapat pada help akan menjadi valid kecuali command "STAR" dan "LOAD". Program ini akan berhenti melakukan looping jika user memberikan input command berupa QUIT.

## 4.2 Pembacaan dan Inisialisasi File Konfigurasi

Dalam pembacaan file konfigurasi dan file data buatan pemain digunakan prosedur dalam ADT mesin karakter yaitu STARTFILE. Prosedur tersebut akan menerima sebuah nama file berupa string dan akan mencoba membuka file dengan nama file yang diberikan pada folder data ke dalam pita. Jika file ditemukan, maka program akan menjalankan prosedur ADV untuk jalan ke karakter setelahnya. Jika file tidak ditemukan atau gagal dibaca, prosedur akan menutup pita.

Lanjutan dari prosedur STARTFILE adalah prosedur STARTWORDFILE yang berada di dalam ADT mesin kata. Prosedur tersebut memiliki algoritma yang mirip dengan primitif ADT mesin kata yaitu STARTWORD, tetapi ia memanggil kata dari STARTFILE dan menyalin seluruh karakter menggunakan prosedur CopyWordWithBlanks (menyalin banyak kata sampai ditemukan ‘\n’).

Kedua prosedur tersebut dan beberapa prosedur lain pada mesin kata dan mesin karakter digunakan didalam prosedur comman START dan LOAD. Prosesnya adalah dengan menjalankan STARTWORDFILE dengan nama file berupa masukan dari user. Setelah dijalankan, program akan mendapatkan jumlah game yang akan dibaca. Program akan men-looping sampai seluruh game pada file terbaca dan telah dimasukkan ke dalam array. Terakhir akan dicek apakah pembacaan berhasil dilakukan dengan melihat apakah isi dari array game lebih dari 0.

## 4.3 Pemanggilan Command

### 4.3.1 START

START merupakan fungsi yang pertama kali dijalankan untuk menyalakan sistem. START akan membaca file konfigurasi default yang berisi list game.

### 4.3.2 LOAD <filename>

LOAD merupakan fungsi yang digunakan untuk membaca suatu save file yang ingin dibuka.

### 4.3.3 SAVE <filename>

SAVE merupakan fungsi yang digunakan untuk menyimpan state game pemain saat ini ke dalam file.

### 4.3.4 CREATE GAME

CREATE GAME merupakan fungsi untuk menambahkan game pada daftar.

### 4.3.5 LIST GAME

LIST GAME merupakan fungsi yang digunakan untuk menampilkan game apa saja yang ada pada program secara terurut, termasuk game buatan user dan game dari sistem.

### 4.3.6 DELETE GAME

DELETE GAME merupakan fungsi yang digunakan untuk menghapus game pada daftar list game. Game yang bisa dihapus hanyalah game yang dibuat secara custom oleh pengguna.

### 4.3.7 QUEUE GAME

QUEUE GAME merupakan fungsi yang mengantrikan game sesuai input user.

### 4.3.8 PLAY GAME

PLAY GAME merupakan fungsi yang akan memainkan game yang ada di kepala antrian. Apabila game yang ingin dimainkan merupakan game tambahan/custom dari pemain, maka akan muncul skor akhir dengan angka yang random.

#### **4.3.9 SKIP GAME <n>**

SKIP GAME akan melewati game sebanyak  $n$  kali dalam antrian.

#### **4.3.10 QUIT**

QUIT merupakan fungsi yang digunakan untuk melakukan *exit* dari program.

#### **4.3.11 HELP**

HELP menampilkan daftar command-command valid yang dapat ditulis pada program.

### **4.4 Game RNG**

Game RNG merupakan permainan yang memanfaatkan fungsi random yang telah dimodifikasi. Dalam program ini, implementasi Game RNG ada di dalam kode `game_rng.c` dan `game_rng.h`. Game RNG akan menghasilkan sebuah angka, dan user diminta untuk menebak. Game akan menyebutkan apabila angka yang disebutkan user lebih besar atau lebih kecil daripada angka yang sedang disimpan.

### **4.5 Game Diner Dash**

Game Diner Dash merupakan permainan yang memanfaatkan ADT queue. Dalam program ini, implementasi Diner Dash ada di dalam kode `dinerdash.c`, `dinerdash.h`, dan menggunakan ADT queue khusus untuk Diner Dash yaitu `q_dinerdash.c` dan `q_dinerdash.h`. Selain itu, Diner Dash juga membaca command dari user menggunakan mesin kata dan mesin karakter.

Pemain akan memasuki permainan Diner Dash dengan state saldo masih sama dengan 0. Saat pertama kali memainkan Diner Dash, terdapat tiga pesanan yang dari sistem. Setiap pesanan memiliki ID pesanan, harga pesanan, durasi memasak, dan durasi ketahanan. Pada setiap putaran, makanan yang sedang dimasak atau sudah disajikan akan berkurang durasi memasak dan ketahanannya. User memasak makanan dari daftar pesanan dengan menuliskan 'COOK <nama makanan>'. Apabila ada makanan tersebut pada antrian, maka makanan akan dimasak. Setelah durasi masakan berkurang dan mencapai 0, maka makanan akan masuk ke daftar penyajian makanan. Makanan harus disajikan dengan mengetik 'SERVE <nama makanan>'. Apabila sudah disajikan, makanan tersebut akan dihapus dari daftar pesanan. Pemain juga bisa melakukan 'SKIP' yang akan menambahkan putaran baru pada permainan tanpa melakukan action yang berarti. Pemain akan mendapat uang seharga harga makanan yang disajikan.

Permainan berakhir ketika jumlah antrian mencapai tujuh dan/atau jumlah makanan yang disajikan sudah lima belas.

## 4.6 Game 2048

Game 2048 merupakan permainan yang memanfaatkan ADT matriks yang telah dimodifikasi. Dalam program ini, implementasi 2048 ada di dalam kode `game_2048.h` dan `game_2048.c`. Game 2048 akan membuat suatu *grid* matriks sebagai papan permainan. Pada setiap awal permainan, akan tercipta suatu angka 2 atau 4 pada *grid* sebagai penanda awal permainan. User kemudian bisa mengetik W (atas), A (kiri), S (bawah), dan D (kanan) untuk menggeser angka-angka pada grid ke atas, kiri, bawah, atau kanan. Apabila ada angka yang bertabrakan dengan angka lainnya dan kedua angka tersebut bernilai sama, maka angka tersebut akan bergabung dan nilai yang baru merupakan jumlah kedua angka tersebut di tempat yang baru.

## 5 Algoritma-Algoritma Menarik

### 5.1 Force Delete Queue

Prosedur `ForceDeleteAt` digunakan di `q_dinerdash.c`. Pada permainan Diner Dash, digunakan ADT queue untuk membuat daftar pesanan, makanan, dan penyajian. Makanan yang telah disajikan akan dihapus dari daftar pesanan, dan seharusnya, makanan di daftar pesanan dihapus dengan cara `dequeue`. Akan tetap, ada kalanya makanan selain yang ada di HEAD queue telah disajikan, dan fungsi `dequeue` hanya bisa menghapus makanan yang ada di HEAD queue. Oleh karena itu, dibuatlah prosedur `ForceDeleteAt` ini agar bisa menghapus elemen di tengah-tengah queue.

### 5.2 Algoritma `scanf()` di C

Ketika sedang mencari tahu bagaimana proses pengambilan input dari user menggunakan mesin kata dan mesin karakter, kami menemukan cara mengimplementasi prosedur `scanf` di dalam C. Ternyata, prosedur `scanf` pada dasarnya mirip dengan mesin karakter yang membaca dari `stdin`. Menariknya adalah cara `scanf` dapat menerima banyak input tanpa harus menspesifikasikan input tersebut sebagai parameter formal pada prosedur. `Scanf` menggunakan sebuah library bernama `stdarg` yang memungkinkan sebuah fungsi/prosedur untuk menerima parameter sebanyak mungkin tanpa harus menspesifikasikan jumlah parameter. Algoritma dari `scanf` juga tidak sulit, sama seperti mesin kata, `scanf` menyalin sebuah kata ke dalam string menggunakan `strcpy()` dan menyalin sebuah angka ke integer menggunakan `atoi()`. Dengan ilmu tersebut, kami dapat membuat `scanf` sendiri yang dapat menerima input berupa command (kata), kalimat (banyak kata), dan integer. Namun, karena keterbatasan library yang dapat digunakan, maksimal parameter dari fungsi kami adalah 3, antara sebuah kalimat saja dan maksimal 2 command dan 1 integer.

## 6 Data Test

Fitur yang dites berupa perintah command yang akan dipakai ketika program berjalan.

## 6.1 START

Fitur yang dites: START
Data tes
<pre>===== BNMO ===== File konfigurasi sistem berhasil dibaca. BNMO berhasil dijalankan.</pre>
Hasil yang seharusnya diberikan
ENTER COMMAND: <b>START</b> File konfigurasi sistem berhasil dibaca. BNMO berhasil dijalankan.

## 6.2 LOAD

Fitur yang dites: LOAD
Data tes: 1. LOAD
<pre>===== BNMO ===== Save file berhasil dibaca. BNMO berhasil dijalankan.</pre>
Hasil yang seharusnya diberikan
ENTER COMMAND: <b>LOAD savefile1.txt</b> Save file berhasil dibaca. BNMO berhasil dijalankan.
Penjelasan lain
File konfigurasi dibaca dan isinya, yang berisikan nama-nama game, dimasukan ke list global

## 6.3 SAVE

Fitur yang dites: SAVE
Data tes:

1. SAVE
<pre>===== BNMO ===== Save file berhasil disimpan.</pre>
Hasil yang seharusnya diberikan
ENTER COMMAND: <b>SAVE</b> Save file berhasil disimpan.

## 6.4 CREATE GAME

Fitur yang dites: CREATE GAME
Data tes: 1. ALSTRUKDAT
<pre>===== BNMO ===== Masukkan nama game yang akan ditambahkan: ALSTRUKDAT Game berhasil ditambahkan</pre>
Hasil yang seharusnya diberikan
ENTER COMMAND: <b>CREATE GAME</b> Masukkan nama game yang akan ditambahkan: <b>ALSTRUKDAT</b> Game berhasil ditambahkan

## 6.5 LIST GAME

Fitur yang dites: LIST GAME
Data tes: 1. LIST GAME
<b>LIST GAME</b>

```

Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. 2048
$
===== ENTER COMMAND :

```

Hasil yang seharusnya diberikan

```

ENTER COMMAND: LIST GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. LUNCH SLOW
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. 2048

```

## 6.6 DELETE GAME

Fitur yang dites: DELETE GAME

Data tes

1. DELETE GAME 2

```

===== BNMO =====
Berikut adalah daftar game yang tersedia
1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. 2048

Masukkan nomor game yang akan dihapus: DELETE GAME 2

Nomor game yang dimasukkan tidak valid.

```

Hasil yang seharusnya diberikan

```

ENTER COMMAND: DELETE GAME

```



Berikut adalah daftar game yang tersedia

1. RNG
2. LUNCH SLOW
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. CUSTOM GAME 1

Masukkan nomor game yang akan dihapus: 6

Game berhasil dihapus

ENTER COMMAND: **DELETE GAME**

Berikut adalah daftar game yang tersedia

1. RNG
2. LUNCH SLOW
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER

Masukkan nomor game yang akan dihapus: 1

Game gagal dihapus

## 6.7 **QUEUE GAME**

Fitur yang dites: QUEUE GAME

Data tes:

1. QUEUE GAME
2. 2
3. 8

===== BNMO =====

Berikut adalah daftar antrian game-mu

Berikut adalah daftar game yang tersedia

1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. 2048

Nomor game yang mau ditambahkan ke antrian: 2

Game berhasil ditambahkan kedalam daftar antrian.

===== BNMO =====

Berikut adalah daftar antrian game-mu

1. Diner DASH

Berikut adalah daftar game yang tersedia

1. RNG
2. Diner DASH
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER
6. 2048

Nomor game yang mau ditambahkan ke antrian: 8

Nomor permainan tidak valid, silahkan masukkan nomor game pada list.

Hasil yang seharusnya diberikan

ENTER COMMAND: **QUEUE GAME**

Berikut adalah daftar antrian game-mu

1. EIFFEL TOWER
2. RISEWOMAN
3. LUNCH SLOW

Berikut adalah daftar game yang tersedia

1. RNG
2. LUNCH SLOW
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER

Nomor Game yang mau ditambahkan ke antrian: 2

Game berhasil ditambahkan ke dalam daftar antrian.

ENTER COMMAND: **QUEUE GAME**

Berikut adalah daftar antrian game-mu

1. EIFFEL TOWER
2. RISEWOMAN
3. LUNCH SLOW

Berikut adalah daftar game yang tersedia

1. RNG
2. LUNCH SLOW
3. DINOSAUR IN EARTH
4. RISEWOMAN
5. EIFFEL TOWER

Nomor Game yang mau ditambahkan ke antrian: **9**

Nomor permainan tidak valid, silahkan masukkan nomor game pada list.

## 6.8 **PLAY GAME**

Fitur yang dites

**PLAY GAME**

Data tes

```
===== BNMO =====  
  
Berikut adalah daftar Game-mu  
1. EIFFEL TOWER  
2. EIFFEL TOWER  
  
Game EIFFEL TOWER masih dalam maintenance, belum dapat dimainkan.  
Silakan pilih game lain.
```

Hasil yang seharusnya diberikan

ENTER COMMAND: **PLAY GAME**

Berikut adalah daftar Game-mu

1. EIFFEL TOWER
2. RISEWOMAN
3. LUNCH SLOW

Loading EIFFEL TOWER ...
<p>ENTER COMMAND: <b>PLAY GAME</b></p> <p>Berikut adalah daftar Game-mu</p> <ol style="list-style-type: none"> <li>1. RISEWOMAN</li> <li>2. RISEWOMAN</li> <li>3. LUNCH SLOW</li> </ol> <p>Game RISEWOMAN masih dalam maintenance, belum dapat dimainkan. Silahkan pilih game lain.</p>

## 6.9 SKIP GAME

Fitur yang dites: SKIP GAME
<p>Data tes:</p> <ol style="list-style-type: none"> <li>1. SKIP GAME 1</li> <li>2. SKIP GAME 3</li> </ol>
<pre> ===== BNMO ===== Berikut adalah daftar Game-mu 1. Diner DASH 2. DINOSAUR IN EARTH 3. NEW GAME? TAPI BOONG:P  ===== BNMO =====  Berikut adalah daftar Game-mu 1. NEW GAME? TAPI BOONG:P  Tidak ada permainan lagi dalam daftar game-mu.  ===== ENTER COMMAND : █ </pre>
Hasil yang seharusnya diberikan
<p>ENTER COMMAND: <b>SKIPGAME 1</b></p> <p>Berikut adalah daftar Game-mu</p> <ol style="list-style-type: none"> <li>1. Diner DASH</li> <li>2. DINOSAUR IN EARTH</li> <li>3. NEW GAME? TAPI BOONG :P</li> </ol> <p>Loading Diner DASH ...</p>

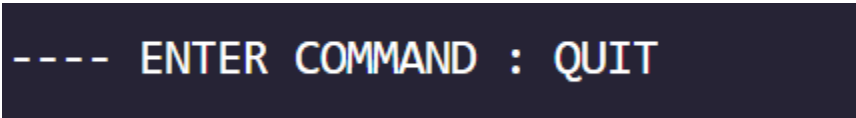
ENTER COMMAND: **SKIP GAME 3**

Tidak ada permainan lagi dalam daftar game-mu.

## 6.10 QUIT

Fitur yang dites: GAME

Data tes: QUIT



```
---- ENTER COMMAND : QUIT
```

Hasil yang seharusnya diberikan

ENTER COMMAND: **QUIT**

Anda keluar dari game BNMO.  
Bye bye ...

## 6.11 HELP

Fitur yang dites: HELP

Data tes

```

===== LIST COMMAND YANG VALID =====
1.  START                : Membaca file konfigurasi sistem.
2.  LOAD <filename>      : Membaca file berisi list game yang dapat dimainkan dan histori.
3.  SAVE <filename>      : Menyimpan state game pada suatu file .txt.
4.  CREATE GAME          : Menambahkan game baru pada daftar game
5.  LIST GAME            : Menampilkan daftar game pada sistem
6.  DELETE GAME          : Menghapus sebuah game dari daftar game.
    (*) Game yang dihapus hanya game yang dibuat pengguna secara custom.
    (*) 5 game pertama pada file konfigurasi tidak dapat dihapus.
    (*) Game yang di dalam queue game saat ini tidak dapat dihapus.
7.  QUEUE GAME           : Mendaftarkan permainan ke dalam list.
8.  PLAY GAME            : Memainkan game dengan urutan pertama di antrian.
9.  SKIP GAME <n>        : Melewatkan permainan sebanyak <n>.
10. QUIT                 : Keluar dari program.
11. HELP                 : Menampilkan list ini.

```

Hasil yang seharusnya diberikan

Bantuan command-command yang disebutkan. Tampilan dan kata-kata dibebaskan.

## 7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur START	Memastikan apakah game console dapat berjalan dan file konfigurasi telah dibaca	Memasukkan command START	START	1. Program membaca file config.txt 2. Program menampilkan pesan "File konfigurasi sistem telah dibaca. BNMO siap dijalankan."	Sesuai yang diharapkan
2	Fitur LOAD	Memastikan apakah file konfigurasi berhasil dibaca dan disimpan isinya di dalam array global	Memasukkan perintah LOAD test1.txt	LOAD test1.txt	1. Program membaca file test1.txt 2. Program menampilkan pesan "Save file berhasil dibaca."	Sesuai yang diharapkan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					BNMO siap dijalankan."	
3	Fitur SAVE	Memastikan berkas konfigurasi tersimpan pada alamat dan nama berkas yang sesuai; dan isi berkas konfigurasi sesuai dengan daftar game	<ol style="list-style-type: none"> <li>1. Membuka program dari direktori akar projek</li> <li>2. Memasukkan perintah START</li> <li>3. Menambahkan game bernama GS</li> <li>4. Mencetak daftar game ke layar</li> <li>5. Memasukkan perintah "SAVE duar.txt"</li> <li>6. Mengakhiri program</li> <li>7. Membuka berkas duar.txt dalam folder data</li> </ol>	<ol style="list-style-type: none"> <li>1. START</li> <li>2. CREATE GAME</li> <li>3. GS</li> <li>4. LIST GAME</li> <li>5. SAVE duar.txt</li> <li>6. QUIT</li> <li>7. cat data/duar.txt</li> </ol>	Berkas duar.txt memiliki isi yang sesuai dengan hasil keluaran LIST GAME pada langkah 4, dengan keterangan jumlah game yang tepat	Sesuai yang diharapkan
4	Fitur CREATE GAME	Memastikan apakah game console dapat ditambahkan game custom dari pengguna	<ol style="list-style-type: none"> <li>1. Memasukkan command CREATE GAME</li> <li>2. Memasukkan nama game tambahan</li> </ol>	<ol style="list-style-type: none"> <li>1. CREATE GAME</li> <li>2. GAMEBARU</li> </ol>	<ol style="list-style-type: none"> <li>1. Program menambahkan game bernama GAMEBARU ke dalam LIST GAME</li> </ol>	Sesuai yang diharapkan
5	Fitur LIST GAME	Memastikan LIST GAME dapat berjalan dengan benar	Memasukkan command LIST GAME	LIST GAME	Program menampilkan daftar game yang tersedia	Sesuai harapan
6	Fitur DELETE GAME	Memastikan game yang memenuhi syarat terhapus dari daftar game, dan game yang tidak dapat dihapus tidak terhapus	<ol style="list-style-type: none"> <li>1. Memulai game dengan perintah START</li> <li>2. Menambahkan game GA dan GB</li> <li>3. Memasukkan game GB ke dalam queue</li> <li>4. Menghapus game GB</li> <li>5. Mencetak daftar game</li> </ol>	<ol style="list-style-type: none"> <li>1. START</li> <li>2. CREATE GAME</li> <li>3. GA</li> <li>4. CREATE GAME</li> <li>5. GB</li> <li>6. QUEUE GAME</li> <li>7. 7</li> <li>8. DELETE GAME</li> <li>9. 7</li> <li>10. LIST GAME</li> </ol>	<ol style="list-style-type: none"> <li>1. Daftar game yang tercetak pada tahap 5 dan 7 menampilkan 8 buah game</li> <li>2. Daftar game yang tercetak pada tahap 9 menampilkan</li> </ol>	Sesuai yang diharapkan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			6. Menghapus game RNG 7. Mencetak daftar game 8. Menghapus game GA 9. Mencetak daftar game	11. DELETE GAME 12. 1 13. LIST GAME 14. DELETE GAME 15. 8 16. LIST GAME	an 7 buah game	
7	Fitur QUEUE GAME	Memastikan game bisa dimasukkan ke dalam daftar antrian permainan yang akan dimainkan	1. Memasukkan command QUEUE GAME 2. Memasukkan game GC ke dalam queue 3. Memasukkan command QUEUE GAME	1. QUEUE GAME 2. GC 3. QUEUE GAME	Daftar game yang tercetak pada tahap 3 menampilkan 1 buah game GC	Sesuai yang diharapkan
8	Fitur PLAY GAME	Memastikan game yang dimainkan adalah permainan pertama yang ada di queue	1. Memasukkan command PLAY GAME 2. Menambahkan antrian game, misalnya 2 3. Memasukkan command PLAY GAME	1. PLAY GAME 2. 2 3. PLAY GAME	Di tahap 1, game tidak dimainkan karena tidak ada game yang masuk ke dalam antrian sistem. Pada tahap 3, game akhirnya bisa dimainkan karena ada game yang masuk ke dalam sistem.	Sesuai yang diharapkan
9	Fitur SKIP GAME	Memastikan antrian game dapat di skip	1. Masuk ke dalam state di mana sudah ada 3 antrian di dalam game 2. Memasukkan command SKIP GAME 1 3. Memasukkan command KIP GAME 3	1. SKIP GAME 1 2. SKIP GAME 3	1. Program menampilkan dua game teratas 2. Program menampilkan tulisan "Tidak ada permainan lagi dalam daftar game-mu."	Sesuai yang diharapkan
10	Fitur QUIT	Memastikan pengguna keluar dari program	Memasukkan perintah QUIT	QUIT	1. Muncul pesan selamat tinggal 2. Program berhenti dijalankan	Sesuai yang diharapkan



No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
11	Fitur HELP	Memastikan bantuan berisi command-c ommand yang bisa diinput tercetak pada layar	1. Memasukkan command HELP	HELP	1. Program mencetak list command-c ommand yang bisa diinput tercetak pada layar	Sesuai yang diharapkan
12	COMMAN D LAIN	Memastikan command lain tidak akan menyebabkan program berjalan ( <i>invalid input</i> )	1. Memasukkan command SEMBARANG	SEMBARANG	1. Program meminta user untuk memasukkan input lagi (balik ke menu/state sebelumnya)	Sesuai yang diharapkan
13	Fitur GAME RNG	Memastikan game RNG dapat dimainkan	1. Masuk ke dalam game 2. Memasukkan nilai 50 3. Memasukkan nilai 75 4. Memasukkan nilai 87 5. Memasukkan nilai 101 6. Memasukkan nilai 95 7. Memasukkan nilai 99 8. Memasukkan nilai 97 9. Memasukkan nilai 98	50 75 87 101 95 99 97 98	1. Game RNG nyala 2. Output: Lebih besar 3. Output: Lebih besar 4. Output: Lebih kecil 5. Output: Lebih besar 6. Output: Lebih kecil 7. Output: Lebih besar 8. Output: Skor-mu adalah 65	Sesuai yang diharapkan
14	Fitur GAME Diner Dash	Memastikan permainan Diner Dash dapat berjalan dengan baik	1. Memasukkan command COOK, SERVE, atau SKIP sampai Game Over 2. Game Over adalah saat antrian pesanan lebih dari 7 atau jumlah makanan yang telah dipesan	COOK M2 COOK M1 COOK M4 COOK M5 SERVE M2	1. Makanan M2 di-enqueue ke dalam queue Cooked 2. Makanan M1, M4, dan M5 di-enqueue satu-satu setiap babak	Sesuai yang diharapkan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			telah mencapai 15		3. Makanan M2 telah selesai dimasak dan memasuki queue serve. 4. Masakan M2 disajikan/di-dequeue.	

## 8 Pembagian Kerja dalam Kelompok

No.	Fitur/ADT	NIM Coder	NIM Tester
1.	ADT Array	18221067, 18221101	18221067, 18221101, 18221163
2.	ADT Matriks	18221067	18221067
3.	ADT Mesin Kata dan Mesin Karakter	18221067	18221067
4.	ADT Queue	18221067	18221067
5.	ADT q_dinerdash	18221101	18221101
6.	Main Menu	18221087	18221087
7.	Game RNG	18221067	18221067
8.	Game Diner Dash	18221101	18221101
9.	Game 2048	18221067	18221067
10.	START	18221067	18221067
11.	LOAD	18221069	18221069
12.	SAVE	18221163	18221163
13.	CREATEGAME	18221101	18221101
14.	LISTGAME	18221087	18221087
15.	DELETGAME	18221163	18221163
16.	QUEUEGAME	18221067	18221067
17.	PLAYGAME	18221069	18221069
19.	SKIPGAME	18221069	18221069
19.	QUIT	18221163	18221163

20.	HELP	18221101	18221101
-----	------	----------	----------

## 9 Lampiran

### 9.1 Deskripsi Tugas Besar 1

BNMO (dibaca: Binomo) adalah sebuah robot video game console yang dimiliki oleh Indra dan Doni. Dua bulan yang lalu, ia mengalami kerusakan dan telah berhasil diperbaiki. Sayangnya, setelah diperbaiki ia justru mendapatkan lebih banyak bug dalam sistemnya. Oleh karena itu, Indra dan Doni mencari programmer lain yang lebih handal untuk ulang memprogram robot video game console kesayangannya.

### 9.2 Notulen Rapat

**Rapat Perdana - 29/10/2022 19.00 , Zoom**

tipe gamelist < list: array of char\*, neff: int >

tipe scoreboard < board: array of score, neff: int >

tipe score < nama: char\*, skor: int >

—

struct gamelist < list: array of char\*, neff: int >

[tinggal how-to ngumpulin/nyusun scoreboard2 tiap game]

struct scoreboard < gamename: char\*, list: score\*, neff: int >

struct score < playername: char\*,score: int >

— ADT —

score.h

scoreboard.h

gamelist.h

queue.h

array.h

— game —

dinnerdash.h

rng.h

custom\_game1.h

other.h { ini buat yg return score random val }

— functionality —

menu.h

main.c { ngurus start dan load }

Pembagian Tugas

1. Ilma - Game Dinnner dash

2. AUFAR - ADT Game list
3. WILLY - ADT Score
4. GIBRAN - ADT Scoreboard
5. GAME RNG - FAWWAZ

### 9.3 Log Activity Anggota Kelompok

Waktu	NIM	Keterangan
29-10-2022 19:00	18221067, 18221069, 18221087, 18221101, 18221163	Rapat perdana dan pembagian tugas perdana
02-11-2022 16:00	18221067, 18221069, 18221087, 18221101, 18221163	Pembagian tugas lanjutan
04-11-2022 20:00	18221067, 18221069, 18221087, 18221101, 18221163	Asistensi I
10-11-2022 19:00	18221067, 18221069, 18221087, 18221101, 18221163	Asistensi II

### 9.4 Asistensi

**Form Asistensi Tugas Besar**  
**IF2110/Algoritma dan Struktur Data**  
**Sem. 1 2022/2023**





No. Kelompok/Kelas : 7/K1  
 Nama Kelompok : FGWIA  
 Anggota Kelompok (Nama/NIM) : 1. Fawwaz Abrial Saffa/18221067  
 2. Gibran Fasha Ghazanfar/18221069  
 3. Willy Frans Farel Sijabat/18221087  
 4. Ilmagita Nariswari /18221101  
 5. AUFAR Ramadhan/18221163



Asisten Pembimbing : Kadek Surya Mahardika (13519165)

Asistensi I


Tanggal : 4/11/2022	Catatan Asistensi:
---------------------	--------------------






STEI- ITB	IF2111-TB1-07	Halaman 36 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

<b>Tempat : Zoom Meeting</b>	
<b>Kehadiran Anggota Kelompok:</b> No NIM Tanda tangan  1 18221067   2 18221069   3 18221087   4 18221101   5 18221163	1. Kalo variabel input yang diterima tetep namanya currentWord aja boleh gak? atau harus dimasukin ke variabel dengan nama lain? JAWAB : Boleh 2. Apa ADT yang sebaiknya diterapkan untuk COOK? Array atau Queue? JAWAB : Terserah 3. Kalo bikin fungsi random, cara ngerandomnya boleh apa aja kan? JAWAB : Boleh asal gak dari library lain selain 4. Github kalo awalnya bikin public trus mau di-private bakal ilang ga isinya? JAWAB : kagak, ubah aja langsung ke private

  6	
	<b>Tanda Tangan Asisten:</b>  <b>Kadek Surya Mahardika</b> <b>13519165</b>

#### Asistensi II

<b>Tanggal : 10/11/2022</b>	<b>Catatan Asistensi:</b>  1. Jangan lupa buat driver 2. Perbaiki bug-bug
<b>Tempat : Zoom Meeting</b>	
<b>Kehadiran Anggota Kelompok:</b> No NIM Tanda tangan  1 18221067    2 18221069	

  3 18221087    4 18221101    5 18221163  	
	Tanda Tangan Asisten:  Kadek Surya Mahardika

	<b>13519165</b>
--	-----------------