Mobilita

Generated by Doxygen 1.9.2

Chapter 1

Todo List

Global newState (GameMap m, ItemList todo, ItemList inProgress, ItemStack bag, ItemQueue order)

Implementasi State, termasuk fungsi-fungsi yang mengubah State game, save State, dan reevaluasi State setiap player menjalankan command.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 BooleanMatrix Struct Reference

Matriks bernilai boolean.

`#include <boolean_matrix.h>`

Data Fields

- boolean contents [20][30]
  2D array untuk menyimpan elemen matriks.
- int rowEff
  Banyak baris matriks.
- int colEff
  Banyak kolom matriks.

4.1.1 Detailed Description

Matriks bernilai boolean.

Definition at line 16 of file boolean_matrix.h.

4.1.2 Field Documentation

4.1.2.1 colEff

`int colEff`

Banyak kolom matriks.

Definition at line 29 of file boolean_matrix.h.

4.1.2.2 contents

```
boolean contents[20][30]
```

2D array untuk menyimpan elemen matriks.

Definition at line 21 of file boolean_matrix.h.

4.1.2.3 rowEff

```
int rowEff
```

Banyak baris matriks.

Definition at line 25 of file boolean_matrix.h.

The documentation for this struct was generated from the following file:

- src/models/boolean_matrix.h

4.2 Gadget Struct Reference

Tipe data yang merepresentasikan gadget yang dapat dibeli. Tipe data ini tidak memiliki constructor karena Gadget yang ada selalu sama (tidak ada konstruksi instance gadget pada runtime).

```
#include <gadget.h>
```

Data Fields

- int id

  Identifier gadget yang unik untuk setiap gadget.
- int price

  Harga gadget.
- char * name

  Nama gadget.

4.2.1 Detailed Description

Tipe data yang merepresentasikan gadget yang dapat dibeli. Tipe data ini tidak memiliki constructor karena Gadget yang ada selalu sama (tidak ada konstruksi instance gadget pada runtime).

Definition at line 17 of file gadget.h.

4.2.2 Field Documentation

4.2.2.1 id

```
int id
```

Identifier gadget yang unik untuk setiap gadget.

Definition at line 22 of file gadget.h.

4.2.2.2 name

```
char* name
```

Nama gadget.

Definition at line 30 of file gadget.h.

4.2.2.3 price

```
int price
```

Harga gadget.

Definition at line 26 of file gadget.h.

The documentation for this struct was generated from the following file:

- src/models/gadget.h

4.3 GadgetList Struct Reference

List statik berisi tepat 5 Gadget.

```
#include <gadget_list.h>
```

Data Fields

- Gadget contents [5]
  Array statik dengan panjang 5 berisi Gadget.

4.3.1 Detailed Description

List statik berisi tepat 5 Gadget.

Definition at line 16 of file gadget_list.h.

4.3.2 Field Documentation

4.3.2.1 contents

```
Gadget contents[5]
```

Array statik dengan panjang 5 berisi Gadget.

Definition at line 21 of file gadget_list.h.

The documentation for this struct was generated from the following file:

- src/models/gadget_list.h

4.4 GameMap Struct Reference

Tipe data berisi ukuran map, matriks adjacency, list lokasi, dan matriks lokasi.

```
#include <game_map.h>
```

Data Fields

- int hSize
- int vSize
- BooleanMatrix _adjacency
- LocationList _locations
- LocationMatrix _locationMatrix

4.4.1 Detailed Description

Tipe data berisi ukuran map, matriks adjacency, list lokasi, dan matriks lokasi.

Definition at line 19 of file game_map.h.

4.4.2 Field Documentation

4.4.2.1 _adjacency

```
BooleanMatrix _adjacency
```

Definition at line 23 of file game_map.h.

4.4.2.2 _locationMatrix

```
LocationMatrix _locationMatrix
```

Definition at line 25 of file game_map.h.

4.4.2.3 _locations

```
LocationList _locations
```

Definition at line 24 of file game_map.h.

4.4.2.4 hSize

```
int hSize
```

Definition at line 21 of file game_map.h.

4.4.2.5 vSize

```
int vSize
```

Definition at line 22 of file game_map.h.

The documentation for this struct was generated from the following file:

- src/models/game_map.h

4.5 Item Struct Reference

Struktur tipe data Item dan pesanan.

```
#include <item.h>
```

Data Fields

- int orderTime
  Waktu pesanan item.
- Location pickUpLocation
  Tempat pick up item.
- Location dropOffLocation
  Tempat drop off item.
- ItemType type
  Tipe item.
- int perishTime
  Waktu hangus item.

4.5.1 Detailed Description

Struktur tipe data Item dan pesanan.

Definition at line 42 of file item.h.

4.5.2 Field Documentation

4.5.2.1 dropOffLocation

```
Location dropOffLocation
```

Tempat drop off item.

Definition at line 55 of file item.h.

4.5.2.2 orderTime

```
int orderTime
```

Waktu pesanan item.

Definition at line 47 of file item.h.

4.5.2.3 perishTime

```
int perishTime
```

Waktu hangus item.

Definition at line 63 of file item.h.

4.5.2.4 pickUpLocation

```
Location pickUpLocation
```

Tempat pick up item.

Definition at line 51 of file item.h.

4.5.2.5 type

```
ItemType type
```

Tipe item.

Definition at line 59 of file item.h.

The documentation for this struct was generated from the following file:

- src/models/item.h

4.6 ItemListNode Struct Reference

Node dari tipe data linked list ItemList.

```
#include <item_list.h>
```

Data Fields

- Item value
  Nilai Item pada ItemListNode ini.
- ItemListNode * next
  Pointer ke ItemListNode selanjutnya.

4.6.1 Detailed Description

Node dari tipe data linked list ItemList.

Definition at line 21 of file item_list.h.

4.6.2 Field Documentation

4.6.2.1 next

```
ItemListNode* next
```

Pointer ke ItemListNode selanjutnya.

Definition at line 30 of file item_list.h.

4.6.2.2 value

```
Item value
```

Nilai Item pada ItemListNode ini.

Definition at line 26 of file item_list.h.

The documentation for this struct was generated from the following file:

- src/models/item_list.h

4.7 ItemQueue Struct Reference

Antrian Item terurut berdasarkan waktu pesanan masuk.

```
#include <item_queue.h>
```

Data Fields

- int headIndex
  Indeks terdepan antrian.
- int tailIndex
  Indeks terakhir antrian.
- Item buffer [30]
  Array tempat menyimpan elemen antrian.

4.7.1 Detailed Description

Antrian Item terurut berdasarkan waktu pesanan masuk.

Definition at line 17 of file item_queue.h.

4.7.2 Field Documentation

4.7.2.1 buffer

```
Item buffer[30]
```

Array tempat menyimpan elemen antrian.

Definition at line 31 of file item_queue.h.

4.7.2.2 headIndex

```
int headIndex
```

Indeks terdepan antrian.

Definition at line 22 of file item_queue.h.

4.7.2.3 tailIndex

```
int tailIndex
```

Indeks terakhir antrian.

Definition at line 26 of file item_queue.h.

The documentation for this struct was generated from the following file:

- src/models/item_queue.h

4.8 ItemStack Struct Reference

Tumpukan Item pada tas.

```
#include <item_stack.h>
```

Data Fields

- int topIndex
  Indeks teratas stack.
- int capacity
  Kapasitas stack.
- Item buffer [100]
  Array tempat menyimpan elemen stack.

4.8.1 Detailed Description

Tumpukan Item pada tas.

Definition at line 16 of file item_stack.h.

4.8.2 Field Documentation

4.8.2.1 buffer

```
Item buffer[100]
```

Array tempat menyimpan elemen stack.

Definition at line 29 of file item_stack.h.

4.8.2.2 capacity

```
int capacity
```

Kapasitas stack.

Definition at line 25 of file item_stack.h.

4.8.2.3 topIndex

```
int topIndex
```

Indeks teratas stack.

Definition at line 21 of file item_stack.h.

The documentation for this struct was generated from the following file:

- src/models/item_stack.h

4.9 Location Struct Reference

Struktur tipe data lokasi yang memuat koordinat, simbol, dan id.

```
#include <location.h>
```

Data Fields

- int id
  Identifier lokasi. ! id harus unik untuk lokasi yang berbeda.
- char symbol
  Simbol lokasi yang dapat ditampilkan.
- Point coordinate
  Koordinat lokasi.
- boolean isPlayerPlace
  Flag yang menandakan apakah lokasi ini sedang ditempati player.
- boolean isPickUpPlace
  Flag yang menandakan apakah lokasi ini adalah lokasi pick up item.
- boolean isDropOffPlace
  Flag yang menandakan apakah lokasi ini adalah lokasi drop off item.
- boolean isReachable
  Flag yang menandakan apakah lokasi ini dapat dituju relatif dari lokasi player.

4.9.1 Detailed Description

Struktur tipe data lokasi yang memuat koordinat, simbol, dan id.

Definition at line 22 of file location.h.

4.9.2 Field Documentation

4.9.2.1 coordinate

```
Point coordinate
```

Koordinat lokasi.

Definition at line 36 of file location.h.

4.9.2.2 id

```
int id
```

Identifier lokasi. ! id harus unik untuk lokasi yang berbeda.

Definition at line 28 of file location.h.

4.9.2.3 isDropOffPlace

```
boolean isDropOffPlace
```

Flag yang menandakan apakah lokasi ini adalah lokasi drop off item.

Definition at line 51 of file location.h.

4.9.2.4 isPickUpPlace

```
boolean isPickUpPlace
```

Flag yang menandakan apakah lokasi ini adalah lokasi pick up item.

Definition at line 46 of file location.h.

4.9.2.5 isPlayerPlace

```
boolean isPlayerPlace
```

Flag yang menandakan apakah lokasi ini sedang ditempati player.

Definition at line 41 of file location.h.

4.9.2.6 isReachable

```
boolean isReachable
```

Flag yang menandakan apakah lokasi ini dapat dituju relatif dari lokasi player.

Definition at line 56 of file location.h.

4.9.2.7 symbol

```
char symbol
```

Simbol lokasi yang dapat ditampilkan.

Definition at line 32 of file location.h.

The documentation for this struct was generated from the following file:

- src/models/location.h

4.10 LocationList Struct Reference

List dinamis berisi data Location.

```
#include <location_list.h>
```

Data Fields

- Location * buffer

  Memory tempat menyimpan elemen list.
- int nEff

  Banyak elemen list.
- int capacity

  Kapasitas list.

4.10.1 Detailed Description

List dinamis berisi data Location.

Definition at line 17 of file location_list.h.

4.10.2 Field Documentation

4.10.2.1 buffer

```
Location* buffer
```

Memory tempat menyimpan elemen list.

Definition at line 22 of file location_list.h.

4.10.2.2 capacity

```
int capacity
```

Kapasitas list.

Definition at line 30 of file location_list.h.

4.10.2.3 nEff

```
int nEff
```

Banyak elemen list.

Definition at line 26 of file location_list.h.

The documentation for this struct was generated from the following file:

- src/models/location_list.h

4.11 LocationMatrix Struct Reference

Matriks berisi data Location.

```
#include <location_matrix.h>
```

Data Fields

- Location contents [20][30]

  2D array untuk menyimpan elemen matriks.
- int rowEff

  Banyak baris matriks yang terdefinisi.
- int colEff

  Banyak kolom matriks yang terdefinisi.

4.11.1 Detailed Description

Matriks berisi data Location.

Definition at line 16 of file location_matrix.h.

4.11.2 Field Documentation

4.11.2.1 colEff

```
int colEff
```

Banyak kolom matriks yang terdefinisi.

Definition at line 29 of file location_matrix.h.

4.11.2.2 contents

```
Location contents[20][30]
```

2D array untuk menyimpan elemen matriks.

Definition at line 21 of file location_matrix.h.

4.11.2.3 rowEff

```
int rowEff
```

Banyak baris matriks yang terdefinisi.

Definition at line 25 of file location_matrix.h.

The documentation for this struct was generated from the following file:

- src/models/location_matrix.h

4.12 Point Struct Reference

Struktur tipe data titik.

`#include <point.h>`

Data Fields

- int x

  Absis suatu titik.
- int y

  Ordinat suatu titik.

4.12.1 Detailed Description

Struktur tipe data titik.

Definition at line 15 of file point.h.

4.12.2 Field Documentation

4.12.2.1 x

`int x`

Absis suatu titik.

Definition at line 20 of file point.h.

4.12.2.2 y

`int y`

Ordinat suatu titik.

Definition at line 24 of file point.h.

The documentation for this struct was generated from the following file:

- src/models/point.h

4.13 State Struct Reference

Game state & life cycle.

`#include <state.h>`

Data Fields

- GameMap gameMap
  Map dari game yang berjalan.
- ItemList todoList
  ToDo List dari game yang berjalan.
- ItemList inProgressList
  In Progress List dari game yang berjalan.
- ItemStack bag
  Tas player.
- ItemQueue order
  Daftar pesanan dari game yang berjalan.
- GadgetList inventory
  Inventory player.
- int cash
  Uang player.
- Location currentLocation
  Lokasi player saat ini.

4.13.1 Detailed Description

Game state & life cycle.

Definition at line 18 of file state.h.

4.13.2 Field Documentation

4.13.2.1 bag

`ItemStack bag`

Tas player.

Definition at line 35 of file state.h.

4.13.2.2 cash

```
int cash
```

Uang player.

Definition at line 47 of file state.h.

4.13.2.3 currentLocation

```
Location currentLocation
```

Lokasi player saat ini.

Definition at line 51 of file state.h.

4.13.2.4 gameMap

```
GameMap gameMap
```

Map dari game yang berjalan.

Definition at line 23 of file state.h.

4.13.2.5 inProgressList

```
ItemList inProgressList
```

In Progress List dari game yang berjalan.

Definition at line 31 of file state.h.

4.13.2.6 inventory

```
GadgetList inventory
```

Inventory player.

Definition at line 43 of file state.h.

4.13.2.7 order

```
ItemQueue order
```

Daftar pesanan dari game yang berjalan.

Definition at line 39 of file state.h.

4.13.2.8 todoList

```
ItemList todoList
```

ToDo List dari game yang berjalan.

Definition at line 27 of file state.h.

The documentation for this struct was generated from the following file:

- src/models/state.h

Chapter 5

File Documentation

5.1 src/models/boolean.h File Reference

Definisi tipe data boolean.

Macros

- #define boolean unsigned char

  Tipe boolean.
- #define true 1

  Representasi nilai `true` pada tipe boolean.
- #define false 0

  Representasi nilai `false` pada tipe boolean.

5.1.1 Detailed Description

Definisi tipe data boolean.

Definition in file boolean.h.

5.1.2 Macro Definition Documentation

5.1.2.1 boolean

```
#define boolean unsigned char
```

Tipe boolean.

Definition at line 13 of file boolean.h.

5.1.2.2 false

```
#define false 0
```

Representasi nilai `false` pada tipe boolean.

Definition at line 23 of file boolean.h.

5.1.2.3 true

```
#define true 1
```

Representasi nilai `true` pada tipe boolean.

Definition at line 18 of file boolean.h.

5.2 boolean.h

Go to the documentation of this file.

```
00001
00006 #ifndef BOOLEAN_H
00007 #define BOOLEAN_H
00008
00013 #define boolean unsigned char
00018 #define true 1
00023 #define false 0
00024
00025 #endif
```

5.3 src/models/boolean_matrix.c File Reference

Implementasi tipe data BooleanMatrix.

```
#include "boolean.h"
#include "boolean_matrix.h"
```

Functions

- BooleanMatrix newBooleanMatrix (int rows, int cols)

  Constructor untuk membuat BooleanMatrix baru.

5.3.1 Detailed Description

Implementasi tipe data BooleanMatrix.

Digunakan untuk matriks adjacency pada instance GameMap.

See also

GameMap

Definition in file boolean_matrix.c.

5.3.2 Function Documentation

5.3.2.1 newBooleanMatrix()

```
BooleanMatrix newBooleanMatrix (
            int rows,
            int cols )
```

Constructor untuk membuat BooleanMatrix baru.

Parameters

rows	Banyak baris efektif.
cols	Banyak kolom efektif.

Returns

`BooleanMatrix` instance.

Definition at line 19 of file boolean_matrix.c.

5.4 boolean_matrix.c

Go to the documentation of this file.

```
00001
00009 #include "boolean.h"
00010 #include "boolean_matrix.h"
00011
00019 BooleanMatrix newBooleanMatrix(int rows, int cols)
00020 {
00021     BooleanMatrix b;
00022     rows(b) = rows;
00023     cols(b) = cols;
00024
```

```
00025     for (int i = 0; i < rows(b); i++)
00026     {
00027         for (int j = 0; j < cols(b); j++)
00028         {
00029             elem(b, i, j) = false;
00030         }
00031     }
00032 
00033     return b;
00034 }
```

5.5 src/models/boolean_matrix.h File Reference

Header file untuk tipe data BooleanMatrix.

`#include "boolean.h"`

Data Structures

- struct BooleanMatrix

  Matriks bernilai boolean.

Macros

- #define rows(b) (b).rowEff

  Mengembalikan banyak baris efektif BooleanMatrix `b`.
- #define cols(b) (b).colEff

  Mengembalikan banyak kolom efektif BooleanMatrix `b`.
- #define elem(b, i, j) (b).contents[i][j]

  Mengembalikan elemen BooleanMatrix `b` pada index (i, j).

Functions

- BooleanMatrix newBooleanMatrix (int rows, int cols)

  Constructor untuk membuat BooleanMatrix baru.

5.5.1 Detailed Description

Header file untuk tipe data BooleanMatrix.

Author

your name (you@domain.com)

Definition in file boolean_matrix.h.

5.5.2 Macro Definition Documentation

5.5.2.1 cols

```
#define cols(
             b ) (b).colEff
```

Mengembalikan banyak kolom efektif BooleanMatrix `b`.

Parameters

b	BooleanMatrix instance.

Definition at line 41 of file boolean_matrix.h.

5.5.2.2 elem

```
#define elem(
             b,
             i,
             j ) (b).contents[i][j]
```

Mengembalikan elemen BooleanMatrix `b` pada index (i, j).

Parameters

b	BooleanMatrix instance.
i	Index baris elemen yang akan diambil.
j	Index kolom elemen yang akan diambil.

Definition at line 48 of file boolean_matrix.h.

5.5.2.3 rows

```
#define rows(
             b ) (b).rowEff
```

Mengembalikan banyak baris efektif BooleanMatrix `b`.

Parameters

b	BooleanMatrix instance.

Definition at line 36 of file boolean_matrix.h.

5.5.3 Function Documentation

5.5.3.1 newBooleanMatrix()

```
BooleanMatrix newBooleanMatrix (
            int rows,
            int cols )
```

Constructor untuk membuat BooleanMatrix baru.

Parameters

rows	Banyak baris efektif (rowEff).
cols	Banyak kolom efektif (colEff).

Returns

BooleanMatrix instance.

Parameters

rows	Banyak baris efektif.
cols	Banyak kolom efektif.

Returns

`BooleanMatrix` instance.

Definition at line 19 of file boolean_matrix.c.

5.6 boolean_matrix.h

Go to the documentation of this file.

```
00001
00007 #ifndef BOOLEAN_MATRIX_H
00008 #define BOOLEAN_MATRIX_H
00009
00010 #include "boolean.h"
00011
00016 typedef struct
00017 {
00021     boolean contents[20][30];
00025     int rowEff;
00029     int colEff;
00030 } BooleanMatrix;
00031
00036 #define rows(b) (b).rowEff
00041 #define cols(b) (b).colEff
00048 #define elem(b, i, j) (b).contents[i][j]
00049
00057 BooleanMatrix newBooleanMatrix(int rows, int cols);
00058
00059 #endif
```

5.7 src/models/gadget.c File Reference

Implementasi tipe data Gadget.

```
#include "boolean.h"
#include "gadget.h"
```

Functions

- boolean isGadgetIdentical (Gadget gadget1, Gadget gadget2)

  Mengecek apakah dua gadget adalah sama. Pengecekan dilakukan berdasarkan id.

Variables

- const Gadget KAIN_PEMBUNGKUS_WAKTU = {0, 800, "Kain Pembungkus Waktu"}

  Instance Gadget "Kain Pembungkus Waktu".
- const Gadget SENTER_PEMBESAR = {1, 1200, "Senter Pembesar"}

  Instance Gadget "Senter Pembesar".
- const Gadget PINTU_KEMANA_SAJA = {2, 1500, "Pintu Kemana Saja"}

  Instance Gadget "Pintu Kemana Saja".
- const Gadget MESIN_WAKTU = {3, 3000, "Mesin Waktu"}

  Instance Gadget "Mesin Waktu".
- const Gadget SENTER_PENGECIL = {4, 800, "Senter Pengecil"}

  Instance Gadget "Senter Pengecil".
- const Gadget NULL_GADGET = {-1, -1, "-"}

  Instance Gadget yang tidak terdefinisi.

5.7.1 Detailed Description

Implementasi tipe data Gadget.

Definition in file gadget.c.

5.7.2 Function Documentation

5.7.2.1 isGadgetIdentical()

```
boolean isGadgetIdentical (
            Gadget gadget1,
            Gadget gadget2 )
```

Mengecek apakah dua gadget adalah sama. Pengecekan dilakukan berdasarkan id.

Parameters

gadget1	Gadget instance.
gadget2	Gadget isntance.

Returns

true jika kedua Gadget adalah sama, false selainnya.

Definition at line 42 of file gadget.c.

5.7.3 Variable Documentation

5.7.3.1 KAIN_PEMBUNGKUS_WAKTU

```
const Gadget KAIN_PEMBUNGKUS_WAKTU = {0, 800, "Kain Pembungkus Waktu"}
```

Instance Gadget "Kain Pembungkus Waktu".

Definition at line 12 of file gadget.c.

5.7.3.2 MESIN_WAKTU

```
const Gadget MESIN_WAKTU = {3, 3000, "Mesin Waktu"}
```

Instance Gadget "Mesin Waktu".

Definition at line 24 of file gadget.c.

5.7.3.3 NULL_GADGET

```
const Gadget NULL_GADGET = {-1, -1, "-"}
```

Instance Gadget yang tidak terdefinisi.

Definition at line 32 of file gadget.c.

5.7.3.4 PINTU_KEMANA_SAJA

```
const Gadget PINTU_KEMANA_SAJA = {2, 1500, "Pintu Kemana Saja"}
```

Instance Gadget "Pintu Kemana Saja".

Definition at line 20 of file gadget.c.

5.7.3.5 SENTER_PEMBESAR

```
const Gadget SENTER_PEMBESAR = {1, 1200, "Senter Pembesar"}
```

Instance Gadget "Senter Pembesar".

Definition at line 16 of file gadget.c.

5.7.3.6 SENTER_PENGECIL

```
const Gadget SENTER_PENGECIL = {4, 800, "Senter Pengecil"}
```

Instance Gadget "Senter Pengecil".

Definition at line 28 of file gadget.c.

5.8 gadget.c

Go to the documentation of this file.

```
00001
00006 #include "boolean.h"
00007 #include "gadget.h"
00008
00012 const Gadget KAIN_PEMBUNGKUS_WAKTU = {0, 800, "Kain Pembungkus Waktu"};
00016 const Gadget SENTER_PEMBESAR = {1, 1200, "Senter Pembesar"};
00020 const Gadget PINTU_KEMANA_SAJA = {2, 1500, "Pintu Kemana Saja"};
00024 const Gadget MESIN_WAKTU = {3, 3000, "Mesin Waktu"};
00028 const Gadget SENTER_PENGECIL = {4, 800, "Senter Pengecil"};
00032 const Gadget NULL_GADGET = {-1, -1, "-"};
00033
00042 boolean isGadgetIdentical(Gadget gadget1, Gadget gadget2)
00043 {
00044     return id(gadget1) == id(gadget2);
00045 }
```

5.9 src/models/gadget.h File Reference

Header file untuk tipe data Gadget.

```
#include "boolean.h"
```

Data Structures

- struct Gadget

  Tipe data yang merepresentasikan gadget yang dapat dibeli. Tipe data ini tidak memiliki constructor karena Gadget yang ada selalu sama (tidak ada konstruksi instance gadget pada runtime).

Macros

- #define id(g) (g).id

  Mengambil property id dari sebuah Gadget.
- #define price(g) (g).price

  Mengambil property price dari sebuah Gadget.
- #define name(g) (g).name

  Mengambil property name dari sebuah Gadget.

Functions

- boolean isGadgetIdentical (Gadget gadget1, Gadget gadget2)

  Mengecek apakah dua gadget adalah sama. Pengecekan dilakukan berdasarkan id.

Variables

- Gadget KAIN_PEMBUNGKUS_WAKTU

  Instance Gadget "Kain Pembungkus Waktu".
- Gadget SENTER_PEMBESAR

  Instance Gadget "Senter Pembesar".
- Gadget PINTU_KEMANA_SAJA

  Instance Gadget "Pintu Kemana Saja".
- Gadget MESIN_WAKTU

  Instance Gadget "Mesin Waktu".
- Gadget SENTER_PENGECIL

  Instance Gadget "Senter Pengecil".
- Gadget NULL_GADGET

  Instance Gadget yang tidak terdefinisi.

5.9.1 Detailed Description

Header file untuk tipe data Gadget.

Definition in file gadget.h.

5.9.2 Macro Definition Documentation

5.9.2.1 id

```
#define id(
             g ) (g).id
```

Mengambil property id dari sebuah Gadget.

Parameters

g	Gadget instance.

Definition at line 37 of file gadget.h.

5.9.2.2 name

```
#define name(
             g ) (g).name
```

Mengambil property name dari sebuah Gadget.

Parameters

g	Gadget instance.

Definition at line 47 of file gadget.h.

5.9.2.3 price

```
#define price(
              g ) (g).price
```

Mengambil property price dari sebuah Gadget.

Parameters

g	Gadget instance.

Definition at line 42 of file gadget.h.

5.9.3 Function Documentation

5.9.3.1 isGadgetIdentical()

```
boolean isGadgetIdentical (
            Gadget gadget1,
            Gadget gadget2 )
```

Mengecek apakah dua gadget adalah sama. Pengecekan dilakukan berdasarkan id.

Parameters

gadget1	Gadget instance.
gadget2	Gadget isntance.

Returns

true jika kedua Gadget adalah sama, false selainnya.

Definition at line 42 of file gadget.c.

5.9.4 Variable Documentation

5.9.4.1 KAIN_PEMBUNGKUS_WAKTU

```
Gadget KAIN_PEMBUNGKUS_WAKTU [extern]
```

Instance Gadget "Kain Pembungkus Waktu".

Definition at line 12 of file gadget.c.

5.9.4.2 MESIN_WAKTU

```
Gadget MESIN_WAKTU [extern]
```

Instance Gadget "Mesin Waktu".

Definition at line 24 of file gadget.c.

5.9.4.3 NULL_GADGET

```
Gadget NULL_GADGET [extern]
```

Instance Gadget yang tidak terdefinisi.

Definition at line 32 of file gadget.c.

5.9.4.4 PINTU_KEMANA_SAJA

```
Gadget PINTU_KEMANA_SAJA [extern]
```

Instance Gadget "Pintu Kemana Saja".

Definition at line 20 of file gadget.c.

5.9.4.5 SENTER_PEMBESAR

```
Gadget SENTER_PEMBESAR [extern]
```

Instance Gadget "Senter Pembesar".

Definition at line 16 of file gadget.c.

5.9.4.6 SENTER_PENGECIL

```
Gadget SENTER_PENGECIL  [extern]
```

Instance Gadget "Senter Pengecil".

Definition at line 28 of file gadget.c.

5.10 gadget.h

Go to the documentation of this file.

```
00001
00006 #ifndef GADGET_H
00007 #define GADGET_H
00008
00009 #include "boolean.h"
00010
00017 typedef struct
00018 {
00022     int id;
00026     int price;
00030     char *name;
00031 } Gadget;
00032
00037 #define id(g) (g).id
00042 #define price(g) (g).price
00047 #define name(g) (g).name
00048
00052 extern Gadget KAIN_PEMBUNGKUS_WAKTU;
00056 extern Gadget SENTER_PEMBESAR;
00060 extern Gadget PINTU_KEMANA_SAJA;
00064 extern Gadget MESIN_WAKTU;
00068 extern Gadget SENTER_PENGECIL;
00072 extern Gadget NULL_GADGET;
00073
00082 boolean isGadgetIdentical(Gadget gadget1, Gadget gadget2);
00083
00084 #endif
```

5.11 src/models/gadget_list.c File Reference

Implementasi tipe data GadgetList. Tipe data ini digunakan untuk inventory pada game.

```
#include <stdio.h>
#include "boolean.h"
#include "gadget.h"
#include "gadget_list.h"
```

Functions

- GadgetList newGadgetList ()

  Constructor untuk membuat GadgetList baru.
- boolean isGadgetListEmpty (GadgetList gList)

  Mengecek apakah suatu GadgetList kosong atau tidak.
- boolean isGadgetListFull (GadgetList gList)

  Mengecek apakah gadget gList penuh atau tidak.
- Gadget getGadget (GadgetList gList, int index)

  Mengambil Gadget instance dari gList pada indeks index. Mengembalikan NULL_GADGET jika index berada di luar range yang berlaku (0..4).
- void setGadget (GadgetList *gList, int index, Gadget g)

  Set elemen gList pada indeks index menjadi Gadget g.
- void displayGadget (GadgetList gList)

  Menuliskan list Gadget (inventory) ke console output. ! Hanya digunakan untuk command INVENTORY.

5.11.1 Detailed Description

Implementasi tipe data GadgetList. Tipe data ini digunakan untuk inventory pada game.

Definition in file gadget_list.c.

5.11.2 Function Documentation

5.11.2.1 displayGadget()

```
void displayGadget (
            GadgetList gList )
```

Menuliskan list Gadget (inventory) ke console output. ! Hanya digunakan untuk command INVENTORY.

Parameters

gList	GadgetList instance.

Definition at line 97 of file gadget_list.c.

5.11.2.2 getGadget()

```
Gadget getGadget (
            GadgetList gList,
            int index )
```

Mengambil Gadget instance dari gList pada indeks index. Mengembalikan NULL_GADGET jika index berada di luar range yang berlaku (0..4).

Mengambil Gadget instance dari gList pada indeks index.

Parameters

gList	GadgetList instance.
index	Indeks dari Gadget pada gList yang akan diambil.

Returns

Gadget instance pada indeks index di GadgetList gList.

Definition at line 74 of file gadget_list.c.

5.11.2.3 isGadgetListEmpty()

```
boolean isGadgetListEmpty (
            GadgetList gList )
```

Mengecek apakah suatu GadgetList kosong atau tidak.

Parameters

gList	GadgetList instance.

Returns

true jika semua elemen gList adalah gadget yang tidak terdefinsi, false selainnya.

Definition at line 34 of file gadget_list.c.

5.11.2.4 isGadgetListFull()

```
boolean isGadgetListFull (
            GadgetList gList )
```

Mengecek apakah gadget gList penuh atau tidak.

Parameters

gList	GadgetList instance.

Returns

true jika semua elemen gList bukanlah gadget yang tidak terdefinsi, false selainnya.

Definition at line 53 of file gadget_list.c.

5.11.2.5 newGadgetList()

```
GadgetList newGadgetList ( )
```

Constructor untuk membuat GadgetList baru.

Returns

Instance GadgetList berisi 5 Gadget yang tidak terdefinisi.

Definition at line 17 of file gadget_list.c.

5.11.2.6 setGadget()

```
void setGadget (
            GadgetList * gList,
            int index,
            Gadget g )
```

Set elemen gList pada indeks index menjadi Gadget g.

Parameters

gList	GadgetList instance.
index	Indeks gList yang akan di-set.
g	Gadget instance.

Definition at line 86 of file gadget_list.c.

5.12 gadget_list.c

Go to the documentation of this file.

```
00001
00007 #include <stdio.h>
00008 #include "boolean.h"
00009 #include "gadget.h"
00010 #include "gadget_list.h"
00011
00017 GadgetList newGadgetList()
00018 {
00019     GadgetList gList;
00020     for (int i = 0; i < 5; i++)
00021     {
00022         setGadget(&gList, i, NULL_GADGET);
00023     }
00024     return gList;
00025 }
00026
00034 boolean isGadgetListEmpty(GadgetList gList)
00035 {
00036     for (int i = 0; i < 5; i++)
00037     {
00038         if (!isGadgetIdentical(getGadget(gList, i), NULL_GADGET))
00039         {
00040             return false;
00041         }
00042     }
00043     return true;
00044 }
00045
00053 boolean isGadgetListFull(GadgetList gList)
00054 {
00055     for (int i = 0; i < 5; i++)
00056     {
00057         if (isGadgetIdentical(getGadget(gList, i), NULL_GADGET))
00058         {
00059             return false;
00060         }
00061     }
00062     return true;
00063 }
00064
00074 Gadget getGadget(GadgetList gList, int index)
00075 {
00076     return (index >= 0 && index < 5) ? gList.contents[index] : NULL_GADGET;
00077 }
00078
00086 void setGadget(GadgetList *gList, int index, Gadget g)
00087 {
00088     gList->contents[index] = g;
00089 }
00090
```

```
00097 void displayGadget(GadgetList gList)
00098 {
00099     for (int i = 0; i < 5; i++)
00100     {
00101         printf("%d. %s\n", i + 1, name(getGadget(gList, i)));
00102     }
00103 }
```

5.13 src/models/gadget_list.h File Reference

Header file untuk tipe data GadgetList.

```
#include "boolean.h"
#include "gadget.h"
```

Data Structures

- struct GadgetList

  List statik berisi tepat 5 Gadget.

Functions

- GadgetList newGadgetList ()

  Constructor untuk membuat GadgetList baru.
- boolean isGadgetListEmpty (GadgetList gList)

  Mengecek apakah suatu GadgetList kosong atau tidak.
- boolean isGadgetListFull (GadgetList gList)

  Mengecek apakah gadget gList penuh atau tidak.
- Gadget getGadget (GadgetList gList, int index)

  Mengambil Gadget instance dari gList pada indeks index.
- void setGadget (GadgetList *gList, int index, Gadget g)

  Set elemen gList pada indeks index menjadi Gadget g.
- void displayGadget (GadgetList gList)

  Menuliskan list Gadget (inventory) ke console output. ! Hanya digunakan untuk command INVENTORY.

5.13.1 Detailed Description

Header file untuk tipe data GadgetList.

Definition in file gadget_list.h.

5.13.2 Function Documentation

5.13.2.1 displayGadget()

```
void displayGadget (
            GadgetList gList )
```

Menuliskan list Gadget (inventory) ke console output. ! Hanya digunakan untuk command INVENTORY.

Parameters

gList	GadgetList instance.

Definition at line 97 of file gadget_list.c.

5.13.2.2 getGadget()

```
Gadget getGadget (
            GadgetList gList,
            int index )
```

Mengambil Gadget instance dari gList pada indeks index.

Parameters

gList	GadgetList instance.
index	Indeks dari Gadget pada gList yang akan diambil.

Returns

Gadget instance pada indeks index di GadgetList gList.

Mengambil Gadget instance dari gList pada indeks index.

Parameters

gList	GadgetList instance.
index	Indeks dari Gadget pada gList yang akan diambil.

Returns

Gadget instance pada indeks index di GadgetList gList.

Definition at line 74 of file gadget_list.c.

5.13.2.3 isGadgetListEmpty()

```
boolean isGadgetListEmpty (
            GadgetList gList )
```

Mengecek apakah suatu GadgetList kosong atau tidak.

Parameters

gList	GadgetList instance.

Returns

true jika semua elemen gList adalah gadget yang tidak terdefinsi, false selainnya.

Definition at line 34 of file gadget_list.c.

5.13.2.4 isGadgetListFull()

```
boolean isGadgetListFull (
            GadgetList gList )
```

Mengecek apakah gadget gList penuh atau tidak.

Parameters

gList	GadgetList instance.

Returns

true jika semua elemen gList bukanlah gadget yang tidak terdefinsi, false selainnya.

Definition at line 53 of file gadget_list.c.

5.13.2.5 newGadgetList()

```
GadgetList newGadgetList ( )
```

Constructor untuk membuat GadgetList baru.

Returns

Instance GadgetList berisi 5 Gadget yang tidak terdefinisi.

Definition at line 17 of file gadget_list.c.

5.13.2.6 setGadget()

```
void setGadget (
            GadgetList * gList,
            int index,
            Gadget g )
```

Set elemen gList pada indeks index menjadi Gadget g.

Parameters

gList	GadgetList instance.
index	Indeks gList yang akan di-set.
g	Gadget instance.

Definition at line 86 of file gadget_list.c.

5.14 gadget_list.h

Go to the documentation of this file.

```
00001
00006 #ifndef GADGET_LIST_H
00007 #define GADGET_LIST_H
00008
00009 #include "boolean.h"
00010 #include "gadget.h"
00011
00016 typedef struct
00017 {
00021     Gadget contents[5];
00022 } GadgetList;
00023
00029 GadgetList newGadgetList();
00030
00038 boolean isGadgetListEmpty(GadgetList gList);
00039
00047 boolean isGadgetListFull(GadgetList gList);
00048
00056 Gadget getGadget(GadgetList gList, int index);
00057
00065 void setGadget(GadgetList *gList, int index, Gadget g);
00066
00073 void displayGadget(GadgetList gList);
00074
00075 #endif
```

5.15 src/models/game_map.c File Reference

Implementasi tipe data GameMap.

```
#include <stdio.h>
#include "location.h"
#include "boolean_matrix.h"
#include "location_list.h"
#include "location_matrix.h"
#include "game_map.h"
```

Functions

- GameMap newGameMap (int hSize, int vSize, BooleanMatrix adjMatrix, LocationList locations)

  Constructor untuk membuat GameMap baru.
- void displayGameMap (GameMap m)

  Menampilkan map game ke console output.
- void displayAdjacentLocation (GameMap m, Location currentLocation)

  Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

- void _getAdjacentLocations (LocationList *adjLocs, LocationList lList, Location currentLocation, BooleanMatrix adjMatrix)
- boolean isAdjacentTo (GameMap m, Location a, Location b)

  Mengecek apakah suatu lokasi adjacent dengan lokasi lain.
- Location getLocationById (GameMap m, int id)

  Mengambil Location instance berdasarkan id.
- Location getLocationBySymbol (GameMap m, char symbol)

  Mengambil Location instance berdasarkan simbol.
- Location getLocationByCoord (GameMap m, Point p)

  Mengambil Location instance berdasarkan koordinat.

5.15.1 Detailed Description

Implementasi tipe data GameMap.

Definition in file game_map.c.

5.15.2 Function Documentation

5.15.2.1 _getAdjacentLocations()

```
void _getAdjacentLocations (
            LocationList * adjLocs,
            LocationList lList,
            Location currentLocation,
            BooleanMatrix adjMatrix )
```

Definition at line 74 of file game_map.c.

5.15.2.2 displayAdjacentLocation()

```
void displayAdjacentLocation (
            GameMap m,
            Location currentLocation )
```

Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

Parameters

m	GameMap instance.
currentLocation	Lokasi saat ini.

Definition at line 68 of file game_map.c.

5.15.2.3 displayGameMap()

```
void displayGameMap (
            GameMap m )
```

Menampilkan map game ke console output.

Parameters

m	GameMap instance yang akan ditampilkan.

Definition at line 36 of file game_map.c.

5.15.2.4 getLocationByCoord()

```
Location getLocationByCoord (
            GameMap m,
            Point p )
```

Mengambil Location instance berdasarkan koordinat.

Parameters

m	GameMap instance.
p	Koordinat Location.

Returns

Location dengan koordinat p.

Definition at line 132 of file game_map.c.

5.15.2.5 getLocationById()

```
Location getLocationById (
            GameMap m,
            int id )
```

Mengambil Location instance berdasarkan id.

Parameters

m	GameMap instance.
id	Id Location.

Returns

Location dengan id 'id'.

Definition at line 108 of file game_map.c.

5.15.2.6 getLocationBySymbol()

```
Location getLocationBySymbol (
            GameMap m,
            char symbol )
```

Mengambil Location instance berdasarkan simbol.

Parameters

m	GameMap instance
symbol	Simbol Location.

Returns

Location dengan simbol 'symbol'.

Definition at line 120 of file game_map.c.

5.15.2.7 isAdjacentTo()

```
boolean isAdjacentTo (
            GameMap m,
            Location a,
            Location b )
```

Mengecek apakah suatu lokasi adjacent dengan lokasi lain.

Parameters

a	Location pertama.
b	Location kedua.

Returns

true jika a dan b sama, false selainnya.

Definition at line 94 of file game_map.c.

5.15.2.8 newGameMap()

```
GameMap newGameMap (
            int hSize,
            int vSize,
            BooleanMatrix adjMatrix,
            LocationList locations )
```

Constructor untuk membuat GameMap baru.

Parameters

hSize	Panjang map (horizontal).
vSize	Lebar map (vertikal).
adjMatrix	Matriks adjacency dari lokasi-lokasi yang ada.
locations	List lokasi yang ada.

Returns

GameMap baru yang terdefinisi.

Definition at line 22 of file game_map.c.

5.16 game_map.c

Go to the documentation of this file.

```
00001
00006 #include <stdio.h>
00007 #include "location.h"
00008 #include "boolean_matrix.h"
00009 #include "location_list.h"
00010 #include "location_matrix.h"
00011 #include "game_map.h"
00012
00022 GameMap newGameMap(int hSize, int vSize, BooleanMatrix adjMatrix, LocationList locations)
00023 {
00024     GameMap m;
00025     mapLength(m) = hSize;
00026     mapWidth(m) = vSize;
00027     adjMatrix(m) = adjMatrix;
00028     locList(m) = locations;
00029 }
00030
00036 void displayGameMap(GameMap m)
00037 {
00038     for (int i = -1; i < mapWidth(m) + 1; i++)
00039     {
00040         if (i == -1 || i == mapWidth(m))
00041         {
00042             for (int j = 0; j < mapLength(m) + 2; j++)
00043             {
00044                 printf("*");
00045             }
00046         }
00047         else
00048         {
00049             printf("*");
00050             for (int j = 0; j < mapLength(m); j++)
00051             {
00052                 writeLocationSymbol(elem(locMatrix(m), i, j));
00053             }
00054
00055             printf("*");
00056         }
00057         printf("\n");
00058     }
```

```
00059 }
00060
00068 void displayAdjacentLocation(GameMap m, Location currentLocation)
00069 {
00070     LocationList adjLocs;
00071     _getAdjacentLocations(&adjLocs, locList(m), currentLocation, adjMatrix(m));
00072 }
00073
00074 void _getAdjacentLocations(LocationList *adjLocs, LocationList lList, Location currentLocation,
       BooleanMatrix adjMatrix)
00075 {
00076     int i = id(currentLocation);
00077     *adjLocs = newLocationList(26);
00078     for (int j = 0; j < cols(adjMatrix); j++)
00079     {
00080         if (elem(adjMatrix, i, j))
00081         {
00082             insertLast(adjLocs, _getLocationById(lList, j));
00083         }
00084     }
00085 }
00086
00094 boolean isAdjacentTo(GameMap m, Location a, Location b)
00095 {
00096     int idA = id(a);
00097     int idB = id(b);
00098     return elem(adjMatrix(m), idA, idB);
00099 }
00100
00108 Location getLocationById(GameMap m, int id)
00109 {
00110     return _getLocationById(locList(m), id);
00111 }
00112
00120 Location getLocationBySymbol(GameMap m, char symbol)
00121 {
00122     return _getLocationBySymbol(locList(m), symbol);
00123 }
00124
00132 Location getLocationByCoord(GameMap m, Point p)
00133 {
00134     return _getLocationByCoord(locList(m), p);
00135 }
```

5.17 src/models/game_map.h File Reference

Header file untuk tipe data GameMap.

```
#include "location.h"
#include "boolean_matrix.h"
#include "location_list.h"
#include "location_matrix.h"
```

Data Structures

- struct GameMap

  Tipe data berisi ukuran map, matriks adjacency, list lokasi, dan matriks lokasi.

Macros

- #define mapLength(m) (m).hSize

  Mengembalikan panjang GameMap m.
- #define mapWidth(m) (m).vSize

  Mengembalikan lebar GameMap m.
- #define adjMatrix(m) (m)._adjacency

Mengembalikan matriks adjacency dari map `m`.

- #define locList(m) (m)._locations

  Mengembalikan list lokasi yang ada pada `m`.

- #define locMatrix(m) (m)._locationMatrix

  Mengembalikan matriks lokasi dari map `m`. *! Hanya digunakan untuk menampilkan output map.*

Functions

- GameMap newGameMap (int hSize, int vSize, BooleanMatrix adjMatrix, Location *locations)

  Constructor untuk membuat GameMap baru.

- void displayGameMap (GameMap m)

  Menampilkan map game ke console output.

- void displayAdjacentLocation (GameMap m, Location currentLocation)

  Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

- boolean isAdjacentTo (GameMap m, Location a, Location b)

  Mengecek apakah suatu lokasi adjacent dengan lokasi lain.

- Location getLocationById (GameMap m, int id)

  Mengambil Location instance berdasarkan id.

- Location getLocationBySymbol (GameMap m, char symbol)

  Mengambil Location instance berdasarkan simbol.

- Location getLocationByCoord (GameMap m, Point p)

  Mengambil Location instance berdasarkan koordinat.

5.17.1 Detailed Description

Header file untuk tipe data GameMap.

Definition in file game_map.h.

5.17.2 Macro Definition Documentation

5.17.2.1 adjMatrix

```
#define adjMatrix(
             m ) (m)._adjacency
```

Mengembalikan matriks adjacency dari map `m`.

See also

BooleanMatrix

Parameters

m	GameMap instance

Definition at line 55 of file game_map.h.

5.17.2.2 locList

```
#define locList(
              m ) (m)._locations
```

Mengembalikan list lokasi yang ada pada m.

See also

LocationList

Parameters

m	GameMap instance

Definition at line 62 of file game_map.h.

5.17.2.3 locMatrix

```
#define locMatrix(
              m ) (m)._locationMatrix
```

Mengembalikan matriks lokasi dari map m. ! Hanya digunakan untuk menampilkan output map.

See also

LocationMatrix

Parameters

m	GameMap instance

Definition at line 70 of file game_map.h.

5.17.2.4 mapLength

```
#define mapLength(
              m ) (m).hSize
```

Mengembalikan panjang GameMap m.

Parameters

m	GameMap instance.

Definition at line 43 of file game_map.h.

5.17.2.5 mapWidth

```
#define mapWidth(
              m ) (m).vSize
```

Mengembalikan lebar GameMap `m`.

Parameters

m	GameMap instance.

Definition at line 48 of file game_map.h.

5.17.3 Function Documentation

5.17.3.1 displayAdjacentLocation()

```
void displayAdjacentLocation (
             GameMap m,
             Location currentLocation )
```

Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

Parameters

m	GameMap instance.
currentLocation	Lokasi saat ini.

Definition at line 68 of file game_map.c.

5.17.3.2 displayGameMap()

```
void displayGameMap (
             GameMap m )
```

Menampilkan map game ke console output.

Parameters

m	GameMap instance yang akan ditampilkan.

Definition at line 36 of file game_map.c.

5.17.3.3 getLocationByCoord()

```
Location getLocationByCoord (
            GameMap m,
            Point p )
```

Mengambil Location instance berdasarkan koordinat.

Parameters

m	GameMap instance.
p	Koordinat Location.

Returns

Location dengan koordinat p.

Definition at line 132 of file game_map.c.

5.17.3.4 getLocationById()

```
Location getLocationById (
            GameMap m,
            int id )
```

Mengambil Location instance berdasarkan id.

Parameters

m	GameMap instance.
id	Id Location.

Returns

Location dengan id 'id'.

Definition at line 108 of file game_map.c.

5.17.3.5 getLocationBySymbol()

```
Location getLocationBySymbol (
            GameMap m,
            char symbol )
```

Mengambil Location instance berdasarkan simbol.

Parameters

m	GameMap instance
symbol	Simbol Location.

Returns

Location dengan simbol 'symbol'.

Definition at line 120 of file game_map.c.

5.17.3.6 isAdjacentTo()

```
boolean isAdjacentTo (
            GameMap m,
            Location a,
            Location b )
```

Mengecek apakah suatu lokasi adjacent dengan lokasi lain.

Parameters

a	Location pertama.
b	Location kedua.

Returns

true jika a dan b sama, false selainnya.

Definition at line 94 of file game_map.c.

5.17.3.7 newGameMap()

```
GameMap newGameMap (
            int hSize,
            int vSize,
            BooleanMatrix adjMatrix,
            Location * locations )
```

Constructor untuk membuat GameMap baru.

Parameters

hSize	Panjang map (horizontal).
vSize	Lebar map (vertikal).
adjMatrix	Matriks adjacency dari lokasi-lokasi yang ada.
locations	List lokasi yang ada.

Returns

GameMap baru yang terdefinisi.

5.18 game_map.h

Go to the documentation of this file.

```
00001
00006 #ifndef GAMEMAP_H
00007 #define GAMEMAP_H
00008
00009 #include "location.h"
00010 #include "boolean_matrix.h"
00011 #include "location_list.h"
00012 #include "location_matrix.h"
00013
00019 typedef struct
00020 {
00021     int hSize;
00022     int vSize;
00023     BooleanMatrix _adjacency;
00024     LocationList _locations;
00025     LocationMatrix _locationMatrix;
00026 } GameMap;
00027
00037 GameMap newGameMap(int hSize, int vSize, BooleanMatrix adjMatrix, Location *locations);
00038
00043 #define mapLength(m) (m).hSize
00048 #define mapWidth(m) (m).vSize
00055 #define adjMatrix(m) (m)._adjacency
00062 #define locList(m) (m)._locations
00070 #define locMatrix(m) (m)._locationMatrix
00071
00077 void displayGameMap(GameMap m);
00078
00086 void displayAdjacentLocation(GameMap m, Location currentLocation);
00087
00095 boolean isAdjacentTo(GameMap m, Location a, Location b);
00096
00104 Location getLocationById(GameMap m, int id);
00105
00113 Location getLocationBySymbol(GameMap m, char symbol);
00114
00122 Location getLocationByCoord(GameMap m, Point p);
00123
00124 #endif
```

5.19 src/models/item.c File Reference

Implementasi tipe data Item. Digunakan untuk mencatat order, termasuk Todo list dan In Progress list.

```
#include "boolean.h"
#include "location.h"
#include "item.h"
```

Functions

- Item newItem (int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int perishTime)

  Constructor untuk membuat Item baru.
- boolean isItemIdentical (Item item1, Item item2)

  Mengecek apakah dua item sama atau tidak.
- boolean isNormalItem (Item item)

  Mengecek apakah tipe suatu item adalah Normal atau bukan.
- boolean isHeavyItem (Item item)

  Mengecek apakah tipe suatu item adalah Heavy atau bukan.
- boolean isPerishableItem (Item item)

  Mengecek apakah tipe suatu item adalah Perishable atau bukan.
- boolean isVIPItem (Item item)

  Mengecek apakah tipe suatu item adalah VIP atau bukan.

5.19.1 Detailed Description

Implementasi tipe data Item. Digunakan untuk mencatat order, termasuk Todo list dan In Progress list.

Definition in file item.c.

5.19.2 Function Documentation

5.19.2.1 isHeavyItem()

```
boolean isHeavyItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Heavy atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Heavy, false selainnya.

Definition at line 64 of file item.c.

5.19.2.2 isItemIdentical()

```
boolean isItemIdentical (
            Item item1,
            Item item2 )
```

Mengecek apakah dua item sama atau tidak.

Parameters

item1	Item instance.
item2	Item instance.

Returns

true jika kedua item sama, false selainnya.

Definition at line 40 of file item.c.

5.19.2.3 isNormalItem()

```
boolean isNormalItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Normal atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Normal, false selainnya.

Definition at line 52 of file item.c.

5.19.2.4 isPerishableItem()

```
boolean isPerishableItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Perishable atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Perishable, false selainnya.

Definition at line 76 of file item.c.

5.19.2.5 isVIPItem()

```
boolean isVIPItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah VIP atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item VIP, false selainnya.

Definition at line 88 of file item.c.

5.19.2.6 newItem()

```
Item newItem (
            int orderTime,
            Location pickUpLocation,
            Location dropOffLocation,
            ItemType type,
            int perishTime )
```

Constructor untuk membuat Item baru.

Parameters

orderTime	Waktu order item.
pickUpLocation	Lokasi pick up item.
dropOffLocation	Lokasi drop off item.
type	Tipe item.
perishTime	Waktu hangus item.

Returns

Item instance baru.

Definition at line 22 of file item.c.

5.20 item.c

Go to the documentation of this file.

```
00001
00008 #include "boolean.h"
00009 #include "location.h"
00010 #include "item.h"
00011
00022 Item newItem(int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int
       perishTime)
00023 {
00024     Item item;
00025     orderTime(item) = orderTime;
00026     pickUpLoc(item) = pickUpLocation;
00027     dropOffLoc(item) = dropOffLocation;
00028     itemType(item) = type;
00029     perishTime(item) = perishTime;
00030     return item;
00031 }
00032
00040 boolean isItemIdentical(Item item1, Item item2)
00041 {
00042     return isLocationIdentical(pickUpLoc(item1), pickUpLoc(item2)) &&
       isLocationIdentical(dropOffLoc(item1), dropOffLoc(item2)) && itemType(item1) == itemType(item2);
00043 }
00044
00052 boolean isNormalItem(Item item)
00053 {
00054     return itemType(item) == NORMAL;
00055 }
00056
00064 boolean isHeavyItem(Item item)
00065 {
00066     return itemType(item) == HEAVY;
00067 }
00068
00076 boolean isPerishableItem(Item item)
00077 {
00078     return itemType(item) == PERISHABLE;
00079 }
00080
00088 boolean isVIPItem(Item item)
00089 {
00090     return itemType(item) == VIP;
00091 }
```

5.21 src/models/item.h File Reference

Header file untuk tipe data Item.

```
#include "location.h"
```

Data Structures

- struct Item

  Struktur tipe data Item dan pesanan.

Macros

- #define NORMAL 0

  Tipe item Normal.
- #define HEAVY 1

  Tipe item Heavy.
- #define PERISHABLE 2

Tipe item Perishable.

- #define VIP 3

  Tipe item VIP.
- #define UNTIMED -1

  Perish time untuk item yang non-perishable.
- #define pickUpLoc(item) (item).pickUpLocation

  Mengambil lokasi pick up item.
- #define dropOffLoc(item) (item).dropOffLocation

  Mengambil lokasi drop off item.
- #define itemType(item) (item).type

  Mengambil tipe item.
- #define perishTime(item) (item).perishTime

  Mengambil waktu hangus item.
- #define orderTime(item) (item).orderTime

  Mengambil waktu order item.

Typedefs

- typedef int ItemType

  Alias untuk tipe ItemType.

Functions

- Item newItem (int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int perishTime)

  Constructor untuk membuat Item baru.
- boolean isItemIdentical (Item item1, Item item2)

  Mengecek apakah dua item sama atau tidak.
- boolean isNormalItem (Item item)

  Mengecek apakah tipe suatu item adalah Normal atau bukan.
- boolean isHeavyItem (Item item)

  Mengecek apakah tipe suatu item adalah Heavy atau bukan.
- boolean isPerishableItem (Item item)

  Mengecek apakah tipe suatu item adalah Perishable atau bukan.
- boolean isVIPItem (Item item)

  Mengecek apakah tipe suatu item adalah VIP atau bukan.

5.21.1 Detailed Description

Header file untuk tipe data Item.

Definition in file item.h.

5.21.2 Macro Definition Documentation

5.21.2.1 dropOffLoc

```
#define dropOffLoc(
              item ) (item).dropOffLocation
```

Mengambil lokasi drop off item.

Parameters

item	Item instance.

Definition at line 75 of file item.h.

5.21.2.2 HEAVY

```
#define HEAVY 1
```

Tipe item Heavy.

Definition at line 22 of file item.h.

5.21.2.3 itemType

```
#define itemType(
              item ) (item).type
```

Mengambil tipe item.

Parameters

item	Item instance.

Definition at line 80 of file item.h.

5.21.2.4 NORMAL

```
#define NORMAL 0
```

Tipe item Normal.

Definition at line 18 of file item.h.

5.21.2.5 orderTime

```
#define orderTime(
              item ) (item).orderTime
```

Mengambil waktu order item.

Parameters

item	Item instance.

Definition at line 90 of file item.h.

5.21.2.6 PERISHABLE

```
#define PERISHABLE 2
```

Tipe item Perishable.

Definition at line 26 of file item.h.

5.21.2.7 perishTime

```
#define perishTime(
              item ) (item).perishTime
```

Mengambil waktu hangus item.

Parameters

item	Item instance.

Definition at line 85 of file item.h.

5.21.2.8 pickUpLoc

```
#define pickUpLoc(
              item ) (item).pickUpLocation
```

Mengambil lokasi pick up item.

Parameters

item	Item instance.

Definition at line 70 of file item.h.

5.21.2.9 UNTIMED

```
#define UNTIMED -1
```

Perish time untuk item yang non-perishable.

Definition at line 36 of file item.h.

5.21.2.10 VIP

```
#define VIP 3
```

Tipe item VIP.

Definition at line 30 of file item.h.

5.21.3 Typedef Documentation

5.21.3.1 ItemType

```
typedef int ItemType
```

Alias untuk tipe ItemType.

Definition at line 14 of file item.h.

5.21.4 Function Documentation

5.21.4.1 isHeavyItem()

```
boolean isHeavyItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Heavy atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Heavy, false selainnya.

Definition at line 64 of file item.c.

5.21.4.2 isItemIdentical()

```
boolean isItemIdentical (
            Item item1,
            Item item2 )
```

Mengecek apakah dua item sama atau tidak.

Parameters

item1	Item instance.
item2	Item instance.

Returns

true jika kedua item sama, false selainnya.

Definition at line 40 of file item.c.

5.21.4.3 isNormalItem()

```
boolean isNormalItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Normal atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Normal, false selainnya.

Definition at line 52 of file item.c.

5.21.4.4 isPerishableItem()

```
boolean isPerishableItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah Perishable atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item Perishable, false selainnya.

Definition at line 76 of file item.c.

5.21.4.5 isVIPItem()

```
boolean isVIPItem (
            Item item )
```

Mengecek apakah tipe suatu item adalah VIP atau bukan.

Parameters

item	Item instance.

Returns

true jika tipe item VIP, false selainnya.

Definition at line 88 of file item.c.

5.21.4.6 newItem()

```
Item newItem (
            int orderTime,
            Location pickUpLocation,
            Location dropOffLocation,
            ItemType type,
            int perishTime )
```

Constructor untuk membuat Item baru.

Parameters

orderTime	Waktu order item.
pickUpLocation	Lokasi pick up item.
dropOffLocation	Lokasi drop off item.
type	Tipe item.
perishTime	Waktu hangus item.

Returns

Item instance baru.

Definition at line 22 of file item.c.

5.22 item.h

Go to the documentation of this file.

```
00001
00006 #ifndef ITEM_H
00007 #define ITEM_H
00008
00009 #include "location.h"
00010
00014 typedef int ItemType;
00018 #define NORMAL 0
00022 #define HEAVY 1
00026 #define PERISHABLE 2
00030 #define VIP 3
00031
00036 #define UNTIMED -1
00037
00042 typedef struct
00043 {
00047     int orderTime;
00051     Location pickUpLocation;
00055     Location dropOffLocation;
00059     ItemType type;
00063     int perishTime;
00064 } Item;
00065
00070 #define pickUpLoc(item) (item).pickUpLocation
00075 #define dropOffLoc(item) (item).dropOffLocation
00080 #define itemType(item) (item).type
00085 #define perishTime(item) (item).perishTime
00090 #define orderTime(item) (item).orderTime
00091
00102 Item newItem(int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int
       perishTime);
00103
00111 boolean isItemIdentical(Item item1, Item item2);
00112
00120 boolean isNormalItem(Item item);
00121
00129 boolean isHeavyItem(Item item);
00130
00138 boolean isPerishableItem(Item item);
00139
00147 boolean isVIPItem(Item item);
00148
00149 #endif
```

5.23 src/models/item_list.c File Reference

Implementasi tipe data ItemList. Digunakan untuk Todo List dan In Progress List.

```
#include <stdlib.h>
#include "boolean.h"
#include "item.h"
#include "item_list.h"
```

Functions

- ItemList newItemList ()

  Constructor untuk membuat ItemList baru.
- ItemListNode newItemListNode (Item item)

  Constructor untuk membuat ItemListNode baru.
- boolean isItemListEmpty (ItemList iList)

  Mengecek apakah suatu ItemList kosong atau tidak.
- boolean isItemListIndexValid (ItemList iList, int index)

  Mengecek apakah suatu bilangan adalah indeks yang valid untuk iList.
- int itemListLength (ItemList iList)

  Mengembalikan panjang suatu ItemList.
- int indexOfItem (ItemList iList, Item item)

  Mencari indeks pertama kemunculan item pada ItemList.
- Item getItem (ItemList iList, int index)

  Mengambil item pada indeks ke index di iList.
- void setItem (ItemList *iList, int index, Item item)

  Set elemen iList pada indeks index menjadi Item item.
- void insertItemFirst (ItemList *iList, Item item)

  Insert Item di awal list iList.
- void insertItemAt (ItemList *iList, int index, Item item)

  Insert Item di indeks tertentu list iList.
- void insertItemLast (ItemList *iList, Item item)

  Insert Item di akhir list iList.
- void deleteItemFirst (ItemList *iList, Item *item)

  Mengambil & mengapus Item pertama pada iList.
- void deleteItemAt (ItemList *iList, int index, Item *item)

  Mengambil & menghapus Item pada indeks index iList.
- void deleteItemLast (ItemList *iList, Item *item)

  Mengambil & menghapus Item terakhir iList.

5.23.1 Detailed Description

Implementasi tipe data ItemList. Digunakan untuk Todo List dan In Progress List.

Definition in file item_list.c.

5.23.2 Function Documentation

5.23.2.1 deleteItemAt()

```
void deleteItemAt (
            ItemList * iList,
            int index,
            Item * item )
```

Mengambil & menghapus Item pada indeks index iList.

Parameters

	iList	ItemList yang akan dilakukan penghapusan.
	index	Indeks Item yang akan dihapus.
out	*item*	Item pada indeks index.

Definition at line 233 of file item_list.c.

5.23.2.2 deleteItemFirst()

```
void deleteItemFirst (
            ItemList * iList,
            Item * item )
```

Mengambil & mengapus Item pertama pada iList.

Parameters

	iList	ItemList yang akan dihapus nilai pertamanya.
out	*item*	Item di posisi pertama iList.

Definition at line 218 of file item_list.c.

5.23.2.3 deleteItemLast()

```
void deleteItemLast (
            ItemList * iList,
            Item * item )
```

Mengambil & menghapus Item terakhir iList.

Parameters

iList	ItemList instance.
item	Item di posisi terakhir iList.

Definition at line 255 of file item_list.c.

5.23.2.4 getItem()

```
Item getItem (
            ItemList iList,
            int index )
```

Mengambil item pada indeks ke index di iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan diambil nilainya.

Returns

Item pada indeks ke index di iList.

Definition at line 117 of file item_list.c.

5.23.2.5 indexOfItem()

```
int indexOfItem (
            ItemList iList,
            Item item )
```

Mencari indeks pertama kemunculan item pada ItemList.

Parameters

iList	ItemList instance.
item	Item yang akan dicari.

Returns

Index pertama kemunculan item atau -1 jika item tidak ditemukan.

Definition at line 89 of file item_list.c.

5.23.2.6 insertItemAt()

```
void insertItemAt (
            ItemList * iList,
            int index,
            Item item )
```

Insert Item di indeks tertentu list iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan dimasukkan Item.
item	Item instance.

Definition at line 178 of file item_list.c.

5.23.2.7 insertItemFirst()

```
void insertItemFirst (
            ItemList * iList,
            Item item )
```

Insert Item di awal list iList.

Parameters

iList	ItemList instance.
item	Item instance.

Definition at line 163 of file item_list.c.

5.23.2.8 insertItemLast()

```
void insertItemLast (
            ItemList * iList,
            Item item )
```

Insert Item di akhir list iList.

Parameters

iList	ItemList instance.
item	Item instance.

Definition at line 200 of file item_list.c.

5.23.2.9 isItemListEmpty()

```
boolean isItemListEmpty (
            ItemList iList )
```

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

iList	ItemList instance.

Returns

true jika iList kosong, false selainnya.

Definition at line 44 of file item_list.c.

5.23.2.10 isItemListIndexValid()

```
boolean isItemListIndexValid (
            ItemList iList,
            int index )
```

Mengecek apakah suatu bilangan adalah indeks yang valid untuk iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan dicek validitasnya.

Returns

true jika indeks index adalah valid, false selainnya.

Definition at line 57 of file item_list.c.

5.23.2.11 itemListLength()

```
int itemListLength (
            ItemList iList )
```

Mengembalikan panjang suatu ItemList.

Parameters

iList	ItemList instance.

Returns

Panjang dari iList.

Definition at line 68 of file item_list.c.

5.23.2.12 newItemList()

```
ItemList newItemList ( )
```

Constructor untuk membuat ItemList baru.

Returns

ItemList kosong.

Definition at line 17 of file item_list.c.

5.23.2.13 newItemListNode()

```
ItemListNode newItemListNode (
            Item item )
```

Constructor untuk membuat ItemListNode baru.

Parameters

item	Value yang di-hold oleh node ini.

Returns

ItemListNode instance berisi item.

Definition at line 30 of file item_list.c.

5.23.2.14 setItem()

```
void setItem (
            ItemList * iList,
            int index,
            Item item )
```

Set elemen iList pada indeks index menjadi Item item.

Parameters

iList	ItemList instance.
index	Indeks iList yang akan di-set.
item	Item instance.

Definition at line 144 of file item_list.c.

5.24 item_list.c

Go to the documentation of this file.

```
00001
00007 #include <stdlib.h>
00008 #include "boolean.h"
00009 #include "item.h"
00010 #include "item_list.h"
00011
00017 ItemList newItemList()
00018 {
00019     ItemList list;
00020     list = (ItemList)malloc(sizeof(ItemListNode));
00021     list = NULL;
00022 }
00023
00030 ItemListNode newItemListNode(Item item)
00031 {
00032     ItemListNode node;
00033     value(node) = item;
00034     next(node) = NULL;
00035     return node;
00036 }
00037
00044 boolean isItemListEmpty(ItemList iList)
00045 {
00046     return iList == NULL;
00047 }
00048
00057 boolean isItemListIndexValid(ItemList iList, int index)
00058 {
00059     return index >= 0 && index < itemListLength(iList);
00060 }
00061
00068 int itemListLength(ItemList iList)
00069 {
00070     ItemList list;
00071     list = iList;
00072     int count = 0;
00073     while (list != NULL)
00074     {
00075         list = next(*list);
00076         count++;
00077     }
00078     return count;
00079 }
00080
00089 int indexOfItem(ItemList iList, Item item)
00090 {
00091     ItemList list;
00092     list = iList;
00093     int index = 0;
00094     boolean found = false;
00095     while (list != NULL && !found)
00096     {
00097         if (isItemIdentical(value(*list), item))
00098         {
00099             found = true;
00100         }
00101         else
00102         {
00103             index++;
00104             list = next(*list);
00105         }
00106     }
00107     return found ? index : -1;
00108 }
00109
00117 Item getItem(ItemList iList, int index)
00118 {
00119     // if (isItemListIndexValid(iList, index))
00120     // {
00121     ItemList list;
00122     list = iList;
00123     int i = 0;
00124     while (i < index)
00125     {
00126         list = next(*list);
00127         i++;
00128     }
00129     return value(*list);
00130     // }
00131     // else
00132     // {
00133     //     return newItem(NULL_LOCATION, NULL_LOCATION, -1, -1);
```

```
00134     // }
00135 }
00136
00144 void setItem(ItemList *iList, int index, Item item)
00145 {
00146     ItemList list;
00147     list = *iList;
00148     int i = 0;
00149     while (i < index)
00150     {
00151         list = next(*list);
00152         i++;
00153     }
00154     value(*list) = item;
00155 }
00156
00163 void insertItemFirst(ItemList *iList, Item item)
00164 {
00165     ItemList list = iList;
00166     ItemListNode node = newItemListNode(item);
00167     **iList = node;
00168     next(**iList) = list;
00169 }
00170
00178 void insertItemAt(ItemList *iList, int index, Item item)
00179 {
00180     ItemList list, nextList;
00181     list = *iList;
00182     ItemListNode node = newItemListNode(item);
00183     int i = 0;
00184     while (i < index)
00185     {
00186         list = next(*list);
00187         i++;
00188     }
00189     nextList = list;
00190     next(node) = nextList;
00191     next(*list) = &node;
00192 }
00193
00200 void insertItemLast(ItemList *iList, Item item)
00201 {
00202     ItemList list, nextList;
00203     list = *iList;
00204     ItemListNode node = newItemListNode(item);
00205     while (list != NULL)
00206     {
00207         list = next(*list);
00208     }
00209     next(*list) = &node;
00210 }
00211
00218 void deleteItemFirst(ItemList *iList, Item *item)
00219 {
00220     ItemList list = *iList;
00221     *item = value(*list);
00222     *iList = next(*list);
00223     free(list);
00224 }
00225
00233 void deleteItemAt(ItemList *iList, int index, Item *item)
00234 {
00235     ItemList list = *iList;
00236     ItemList nextList;
00237     int i = 0;
00238     while (i < index - 1)
00239     {
00240         list = next(*list);
00241         i++;
00242     }
00243     nextList = next(*list);
00244     *item = value(*nextList);
00245     next(*list) = next(*nextList);
00246     free(nextList);
00247 }
00248
00255 void deleteItemLast(ItemList *iList, Item *item)
00256 {
00257     ItemList list = *iList;
00258     if (next(*list) == NULL)
00259     {
00260         *item = value(*list);
00261         *iList = NULL;
00262         free(list);
00263     }
00264     else
00265     {
```

```
00266         ItemList nextList;
00267         nextList = next(*list);
00268         while (next(*nextList) != NULL)
00269         {
00270             list = nextList;
00271             nextList = next(*nextList);
00272         }
00273         *item = value(*nextList);
00274         next(*list) = NULL;
00275         free(nextList);
00276     }
00277 }
```

5.25 src/models/item_list.h File Reference

Header file untuk tipe data ItemList.

```
#include "boolean.h"
#include "item.h"
```

Data Structures

- struct ItemListNode

  Node dari tipe data linked list ItemList.

Macros

- #define value(node) (node).value

  Mengambil value dari sebuah ItemListNode.
- #define next(node) (node).next

  Mengambil pointer ke next node dari sebuah ItemListNode.

Typedefs

- typedef ItemListNode * ItemList

  Alias untuk tipe data ItemListNode.

Functions

- ItemList newItemList ()

  Constructor untuk membuat ItemList baru.
- ItemListNode newItemListNode (Item item)

  Constructor untuk membuat ItemListNode baru.
- boolean isItemListEmpty (ItemList iList)

  Mengecek apakah suatu ItemList kosong atau tidak.
- boolean isItemListIndexValid (ItemList iList, int index)

  Mengecek apakah suatu bilangan adalah indeks yang valid untuk iList.
- int itemListLength (ItemList iList)

  Mengembalikan panjang suatu ItemList.
- int indexOfItem (ItemList iList, Item item)

Mencari indeks pertama kemunculan item pada ItemList.

- Item getItem (ItemList iList, int index)

  Mengambil item pada indeks ke index di iList.
- void setItem (ItemList *iList, int index, Item item)

  Set elemen iList pada indeks index menjadi Item item.
- void insertItemFirst (ItemList *iList, Item item)

  Insert Item di awal list iList.
- void insertItemAt (ItemList *iList, int index, Item item)

  Insert Item di indeks tertentu list iList.
- void insertItemLast (ItemList *iList, Item item)

  Insert Item di akhir list iList.
- void deleteItemFirst (ItemList *iList, Item *item)

  Mengambil & mengapus Item pertama pada iList.
- void deleteItemAt (ItemList *iList, int index, Item *item)

  Mengambil & menghapus Item pada indeks index iList.
- void deleteItemLast (ItemList *iList, Item *item)

  Mengambil & menghapus Item terakhir iList.

5.25.1 Detailed Description

Header file untuk tipe data ItemList.

Definition in file item_list.h.

5.25.2 Macro Definition Documentation

5.25.2.1 next

```
#define next(
             node ) (node).next
```

Mengambil pointer ke next node dari sebuah ItemListNode.

Parameters

node	ItemListNode instance.

Definition at line 42 of file item_list.h.

5.25.2.2 value

```
#define value(
             node ) (node).value
```

Mengambil value dari sebuah ItemListNode.

Parameters

node	ItemListNode instance.

Definition at line 37 of file item_list.h.

5.25.3 Typedef Documentation

5.25.3.1 ItemList

```
typedef ItemListNode* ItemList
```

Alias untuk tipe data ItemListNode.

Definition at line 15 of file item_list.h.

5.25.4 Function Documentation

5.25.4.1 deleteItemAt()

```
void deleteItemAt (
            ItemList * iList,
            int index,
            Item * item )
```

Mengambil & menghapus Item pada indeks index iList.

Parameters

	iList	ItemList yang akan dilakukan penghapusan.
	index	Indeks Item yang akan dihapus.
out	*item*	Item pada indeks index.

Definition at line 233 of file item_list.c.

5.25.4.2 deleteItemFirst()

```
void deleteItemFirst (
            ItemList * iList,
            Item * item )
```

Mengambil & mengapus Item pertama pada iList.

Parameters

	iList	ItemList yang akan dihapus nilai pertamanya.
out	*item*	Item di posisi pertama iList.

Definition at line 218 of file item_list.c.

5.25.4.3 deleteItemLast()

```
void deleteItemLast (
            ItemList * iList,
            Item * item )
```

Mengambil & menghapus Item terakhir iList.

Parameters

iList	ItemList instance.
item	Item di posisi terakhir iList.

Definition at line 255 of file item_list.c.

5.25.4.4 getItem()

```
Item getItem (
            ItemList iList,
            int index )
```

Mengambil item pada indeks ke index di iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan diambil nilainya.

Returns

Item pada indeks ke index di iList.

Definition at line 117 of file item_list.c.

5.25.4.5 indexOfItem()

```
int indexOfItem (
            ItemList iList,
            Item item )
```

Mencari indeks pertama kemunculan item pada ItemList.

Parameters

iList	ItemList instance.
item	Item yang akan dicari.

Returns

Index pertama kemunculan item atau -1 jika item tidak ditemukan.

Definition at line 89 of file item_list.c.

5.25.4.6 insertItemAt()

```
void insertItemAt (
            ItemList * iList,
            int index,
            Item item )
```

Insert Item di indeks tertentu list iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan dimasukkan Item.
item	Item instance.

Definition at line 178 of file item_list.c.

5.25.4.7 insertItemFirst()

```
void insertItemFirst (
            ItemList * iList,
            Item item )
```

Insert Item di awal list iList.

Parameters

iList	ItemList instance.
item	Item instance.

Definition at line 163 of file item_list.c.

5.25.4.8 insertItemLast()

```
void insertItemLast (
            ItemList * iList,
            Item item )
```

Insert Item di akhir list iList.

Parameters

iList	ItemList instance.
item	Item instance.

Definition at line 200 of file item_list.c.

5.25.4.9 isItemListEmpty()

```
boolean isItemListEmpty (
            ItemList iList )
```

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

iList	ItemList instance.

Returns

true jika iList kosong, false selainnya.

Definition at line 44 of file item_list.c.

5.25.4.10 isItemListIndexValid()

```
boolean isItemListIndexValid (
            ItemList iList,
            int index )
```

Mengecek apakah suatu bilangan adalah indeks yang valid untuk iList.

Parameters

iList	ItemList instance.
index	Indeks yang akan dicek validitasnya.

Returns

true jika indeks index adalah valid, false selainnya.

Definition at line 57 of file item_list.c.

5.25.4.11 itemListLength()

```
int itemListLength (
            ItemList iList )
```

Mengembalikan panjang suatu ItemList.

Parameters

iList	ItemList instance.

Returns

Panjang dari iList.

Definition at line 68 of file item_list.c.

5.25.4.12 newItemList()

```
ItemList newItemList ( )
```

Constructor untuk membuat ItemList baru.

Returns

ItemList kosong.

Definition at line 17 of file item_list.c.

5.25.4.13 newItemListNode()

```
ItemListNode newItemListNode (
            Item item )
```

Constructor untuk membuat ItemListNode baru.

Parameters

item	Value yang di-hold oleh node ini.

Returns

ItemListNode instance berisi item.

Definition at line 30 of file item_list.c.

5.25.4.14 setItem()

```
void setItem (
             ItemList * iList,
             int index,
             Item item )
```

Set elemen iList pada indeks index menjadi Item item.

Parameters

iList	ItemList instance.
index	Indeks iList yang akan di-set.
item	Item instance.

Definition at line 144 of file item_list.c.

5.26 item_list.h

Go to the documentation of this file.

```
00001
00006 #ifndef ITEM_DYNAMIC_LIST_H
00007 #define ITEM_DYNAMIC_LIST_H
00008
00009 #include "boolean.h"
00010 #include "item.h"
00011
00015 typedef ItemListNode *ItemList;
00016
00021 typedef struct
00022 {
00026     Item value;
00030     ItemListNode *next;
00031 } ItemListNode;
00032
00037 #define value(node) (node).value
00042 #define next(node) (node).next
00043
00049 ItemList newItemList();
00050
00057 ItemListNode newItemListNode(Item item);
00058
00065 boolean isItemListEmpty(ItemList iList);
00066
00075 boolean isItemListIndexValid(ItemList iList, int index);
00076
00083 int itemListLength(ItemList iList);
```

```
00084
00093 int indexOfItem(ItemList iList, Item item);
00094
00102 Item getItem(ItemList iList, int index);
00103
00111 void setItem(ItemList *iList, int index, Item item);
00112
00119 void insertItemFirst(ItemList *iList, Item item);
00120
00128 void insertItemAt(ItemList *iList, int index, Item item);
00129
00136 void insertItemLast(ItemList *iList, Item item);
00137
00144 void deleteItemFirst(ItemList *iList, Item *item);
00145
00153 void deleteItemAt(ItemList *iList, int index, Item *item);
00154
00161 void deleteItemLast(ItemList *iList, Item *item);
00162
00163 #endif
```

5.27 src/models/item_queue.c File Reference

Implementasi tipe data ItemQueue. Hanya digunakan untuk antrian pesanan masuk.

```
#include "item.h"
#include "item_queue.h"
```

Functions

- ItemQueue newItemQueue ()

  Constructor untuk membuat ItemQueue baru.
- int peekHeadTime (ItemQueue q)

  Melihat nilai pesanan masuk (orderTime) dari Item terdepan pada q.
- boolean isEmpty (ItemQueue q)

  Mengecek apakah queue q kosong atau tidak.
- void enqueue (ItemQueue *q, Item item)

  Menambah Item item pada antrian q.
- void dequeue (ItemQueue *q, Item *item)

  Mengambil Item terdepan pada antrian q.

5.27.1 Detailed Description

Implementasi tipe data ItemQueue. Hanya digunakan untuk antrian pesanan masuk.

Definition in file item_queue.c.

5.27.2 Function Documentation

5.27.2.1 dequeue()

```
void dequeue (
            ItemQueue * q,
            Item * item )
```

Mengambil Item terdepan pada antrian q.

Parameters

q	ItemQueue instance.
item	Item terdepan pada antrian q.

Definition at line 89 of file item_queue.c.

5.27.2.2 enqueue()

```
void enqueue (
            ItemQueue * q,
            Item item )
```

Menambah Item item pada antrian q.

Parameters

q	ItemQueue instance.
item	Item instance.

Definition at line 52 of file item_queue.c.

5.27.2.3 isEmpty()

```
boolean isEmpty (
            ItemQueue q )
```

Mengecek apakah queue q kosong atau tidak.

Parameters

q	ItemQueue instance yang akan dicek.

Returns

true jika q kosong, false selainnya.

Definition at line 41 of file item_queue.c.

5.27.2.4 newItemQueue()

```
ItemQueue newItemQueue ( )
```

Constructor untuk membuat ItemQueue baru.

Returns

ItemQueue instance baru yang kosong.

Definition at line 15 of file item_queue.c.

5.27.2.5 peekHeadTime()

```
int peekHeadTime (
            ItemQueue q )
```

Melihat nilai pesanan masuk (orderTime) dari Item terdepan pada q.

See also

Item

Parameters

q	ItemQueue instance.

Returns

Nilai pesanan Item terdepan q.

Definition at line 30 of file item_queue.c.

5.28 item_queue.c

Go to the documentation of this file.

```
00001
00007 #include "item.h"
00008 #include "item_queue.h"
00009
00015 ItemQueue newItemQueue()
00016 {
00017     ItemQueue q;
00018     headIndex(q) = -1;
00019     tailIndex(q) = -1;
00020 }
00021
00030 int peekHeadTime(ItemQueue q)
00031 {
00032     return head(q).orderTime;
00033 }
00034
00041 boolean isEmpty(ItemQueue q)
00042 {
00043     return headIndex(q) == -1 && tailIndex(q) == -1;
00044 }
00045
00052 void enqueue(ItemQueue *q, Item item)
00053 {
00054     if (isEmpty(*q))
00055     {
00056         headIndex(*q) = 0;
00057         tailIndex(*q) = 0;
00058         tail(*q) = item;
```

```
00059     }
00060     else
00061     {
00062         // * Insert
00063         tailIndex(*q)++;
00064         tail(*q) = item;
00065         // * Sort
00066         for (int i = 0; i < tailIndex(*q); i++)
00067         {
00068             int priorityIndex = i;
00069             for (int j = i + 1; j <= tailIndex(*q); j++)
00070             {
00071                 if ((*q).buffer[j].orderTime < (*q).buffer[priorityIndex].orderTime)
00072                 {
00073                     priorityIndex = j;
00074                 }
00075             }
00076             Item temp = (*q).buffer[priorityIndex];
00077             (*q).buffer[priorityIndex] = (*q).buffer[i];
00078             (*q).buffer[i] = temp;
00079         }
00080     }
00081 }
00082
00089 void dequeue(ItemQueue *q, Item *item)
00090 {
00091     *item = head(*q);
00092     headIndex(*q)++;
00093 }
```

5.29 src/models/item_queue.h File Reference

Header file untuk tipe data ItemQueue.

```
#include "boolean.h"
#include "item.h"
```

Data Structures

- struct ItemQueue

  Antrian Item terurut berdasarkan waktu pesanan masuk.

Macros

- #define headIndex(q) (q).headIndex

  Mengambil indeks head pada antrian q.
- #define tailIndex(q) (q).tailIndex

  Mengambil indeks tail pada antrian q.
- #define head(q) (q).buffer[(q).headIndex]

  Mengambil head Item pada antrian q.
- #define tail(q) (q).buffer[(q).tailIndex]

  Mengambil tail Item pada antrian q.

Functions

- ItemQueue newItemQueue ()

  Constructor untuk membuat ItemQueue baru.
- int peekHeadTime (ItemQueue q)

  Melihat nilai pesanan masuk (orderTime) dari Item terdepan pada q.
- boolean isEmpty (ItemQueue q)

  Mengecek apakah queue q kosong atau tidak.
- void enqueue (ItemQueue *q, Item item)

  Menambah Item item pada antrian q.
- void dequeue (ItemQueue *q, Item *item)

  Mengambil Item terdepan pada antrian q.

5.29.1 Detailed Description

Header file untuk tipe data ItemQueue.

Definition in file item_queue.h.

5.29.2 Macro Definition Documentation

5.29.2.1 head

```
#define head(
              q ) (q).buffer[(q).headIndex]
```

Mengambil head Item pada antrian q.

Parameters

q	ItemQueue instance.

Definition at line 48 of file item_queue.h.

5.29.2.2 headIndex

```
#define headIndex(
              q ) (q).headIndex
```

Mengambil indeks head pada antrian q.

Parameters

q	ItemQueue instance.

Definition at line 38 of file item_queue.h.

5.29.2.3 tail

```
#define tail(
              q ) (q).buffer[(q).tailIndex]
```

Mengambil tail Item pada antrian q.

Parameters

q	ItemQueue instance.

Definition at line 53 of file item_queue.h.

5.29.2.4 tailIndex

```
#define tailIndex(
              q ) (q).tailIndex
```

Mengambil indeks tail pada antrian q.

Parameters

q	ItemQueue instance.

Definition at line 43 of file item_queue.h.

5.29.3 Function Documentation

5.29.3.1 dequeue()

```
void dequeue (
             ItemQueue * q,
             Item * item )
```

Mengambil Item terdepan pada antrian q.

Parameters

q	ItemQueue instance.
item	Item terdepan pada antrian q.

Definition at line 89 of file item_queue.c.

5.29.3.2 enqueue()

```
void enqueue (
            ItemQueue * q,
            Item item )
```

Menambah Item item pada antrian q.

Parameters

q	ItemQueue instance.
item	Item instance.

Definition at line 52 of file item_queue.c.

5.29.3.3 isEmpty()

```
boolean isEmpty (
            ItemQueue q )
```

Mengecek apakah queue q kosong atau tidak.

Parameters

q	ItemQueue instance yang akan dicek.

Returns

true jika q kosong, false selainnya.

Definition at line 41 of file item_queue.c.

5.29.3.4 newItemQueue()

```
ItemQueue newItemQueue ( )
```

Constructor untuk membuat ItemQueue baru.

Returns

ItemQueue instance baru yang kosong.

Definition at line 15 of file item_queue.c.

5.29.3.5 peekHeadTime()

```
int peekHeadTime (
            ItemQueue q )
```

Melihat nilai pesanan masuk (orderTime) dari Item terdepan pada q.

See also

Item

Parameters

q	ItemQueue instance.

Returns

Nilai pesanan Item terdepan q.

Definition at line 30 of file item_queue.c.

5.30 item_queue.h

Go to the documentation of this file.

```
00001
00006 #ifndef ITEM_QUEUE_H
00007 #define ITEM_QUEUE_H
00008
00009 #include "boolean.h"
00010 #include "item.h"
00011
00017 typedef struct
00018 {
00022     int headIndex;
00026     int tailIndex;
00031     Item buffer[30];
00032 } ItemQueue;
00033
00038 #define headIndex(q) (q).headIndex
00043 #define tailIndex(q) (q).tailIndex
00048 #define head(q) (q).buffer[(q).headIndex]
00053 #define tail(q) (q).buffer[(q).tailIndex]
00054
00060 ItemQueue newItemQueue();
00061
00070 int peekHeadTime(ItemQueue q);
00071
00078 boolean isEmpty(ItemQueue q);
00079
00086 void enqueue(ItemQueue *q, Item item);
00087
00094 void dequeue(ItemQueue *q, Item *item);
00095
00096 #endif
```

5.31 src/models/item_stack.c File Reference

Implementasi tipe data ItemStack. Hanya digunakan untuk INVENTORY.

```
#include "boolean.h"
#include "item.h"
#include "item_stack.h"
```

Functions

- ItemStack newItemStack (int capacity)

  Constructor untuk membuat ItemStack baru.
- boolean isStackEmpty (ItemStack stack)

  Mengecek apakah stack kosong atau tidak.
- boolean isStackFull (ItemStack stack)

  Mengecek apakah stack penuh atau tidak.
- void push (ItemStack *stack, Item item)

  Memasukkan item ke atas stack.
- void pop (ItemStack *stack, Item *item)

  Mengambil item dari atas stack.
- void incrementCapacity (ItemStack *stack)

  Menambah kapasitas stack sebanyak 1. Kapasitas tidak bertambah jika telah mencapai kapasitas maksimum (100).
- void doubleCapacity (ItemStack *stack)

  Menggandakan kapasitas stack. Kapasitas tidak bertambah hingga melebihi kapasitas maksimum (100).
- void _clampCapacity (ItemStack *stack)

  Membatasi kapasitas stack jika kapasitas melebihi batas.

5.31.1 Detailed Description

Implementasi tipe data ItemStack. Hanya digunakan untuk INVENTORY.

Definition in file item_stack.c.

5.31.2 Function Documentation

5.31.2.1 _clampCapacity()

```
void _clampCapacity (
            ItemStack * stack )
```

Membatasi kapasitas stack jika kapasitas melebihi batas.

Parameters

stack	ItemStack instance.

Definition at line 103 of file item_stack.c.

5.31.2.2 doubleCapacity()

```
void doubleCapacity (
            ItemStack * stack )
```

Menggandakan kapasitas stack. Kapasitas tidak bertambah hingga melebihi kapasitas maksimum (100).

Parameters

stack	ItemStack instance.

Definition at line 91 of file item_stack.c.

5.31.2.3 incrementCapacity()

```
void incrementCapacity (
            ItemStack * stack )
```

Menambah kapasitas stack sebanyak 1. Kapasitas tidak bertambah jika telah mencapai kapasitas maksimum (100).

Parameters

stack	ItemStack instance.

Definition at line 78 of file item_stack.c.

5.31.2.4 isStackEmpty()

```
boolean isStackEmpty (
            ItemStack stack )
```

Mengecek apakah stack kosong atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack kosong, false selainnya.

Definition at line 31 of file item_stack.c.

5.31.2.5 isStackFull()

```
boolean isStackFull (
            ItemStack stack )
```

Mengecek apakah stack penuh atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack penuh, false selainnya.

Definition at line 42 of file item_stack.c.

5.31.2.6 newItemStack()

```
ItemStack newItemStack (
            int capacity )
```

Constructor untuk membuat ItemStack baru.

Parameters

capacity	Kapasitas stack.

Returns

ItemStack instace baru yang kosong.

Definition at line 17 of file item_stack.c.

5.31.2.7 pop()

```
void pop (
            ItemStack * stack,
            Item * item )
```

Mengambil item dari atas stack.

Parameters

	stack	ItemStack instance.
out	*item*	Item yang diambil dari atas stack.

Definition at line 65 of file item_stack.c.

5.31.2.8 push()

```
void push (
            ItemStack * stack,
            Item item )
```

Memasukkan item ke atas stack.

Parameters

stack	ItemStack instance.
item	Item yang akan dimasukkan ke atas stack.

Definition at line 53 of file item_stack.c.

5.32 item_stack.c

Go to the documentation of this file.

```
00001
00007 #include "boolean.h"
00008 #include "item.h"
00009 #include "item_stack.h"
00010
00017 ItemStack newItemStack(int capacity)
00018 {
00019     ItemStack s;
00020     topIndex(s) = -1;
00021     capacity(s) = capacity;
00022     return s;
00023 }
00024
00031 boolean isStackEmpty(ItemStack stack)
00032 {
00033     return topIndex(stack) == -1;
00034 }
00035
00042 boolean isStackFull(ItemStack stack)
00043 {
00044     return topIndex(stack) == capacity(stack) - 1;
00045 }
00046
00053 void push(ItemStack *stack, Item item)
00054 {
00055     topIndex(*stack)++;
00056     top(*stack) = item;
00057 }
00058
00065 void pop(ItemStack *stack, Item *item)
00066 {
00067     *item = top(*stack);
00068     topIndex(*stack)--;
00069 }
00070
00078 void incrementCapacity(ItemStack *stack)
```

```
00079 {
00080     capacity(*stack)++;
00081     _clampCapacity(stack);
00082 }
00083
00091 void doubleCapacity(ItemStack *stack)
00092 {
00093     capacity(*stack) *= 2;
00094     _clampCapacity(stack);
00095 }
00096
00103 void _clampCapacity(ItemStack *stack)
00104 {
00105     if (capacity(*stack) > ITEM_STACK_MAX_CAPACITY)
00106     {
00107         capacity(*stack) = ITEM_STACK_MAX_CAPACITY;
00108     }
00109 }
```

5.33 src/models/item_stack.h File Reference

Header file untuk tipe data ItemStack.

```
#include "boolean.h"
#include "item.h"
```

Data Structures

- struct ItemStack

  Tumpukan Item pada tas.

Macros

- #define ITEM_STACK_MAX_CAPACITY 100

  Kapasitas maksimum ItemStack.
- #define topIndex(s) (s).topIndex

  Mengambil indeks teratas stack s.
- #define top(s) (s).buffer[(s).topIndex]

  Mengambil Item teratas pada stack s.
- #define capacity(s) (s).capacity

  Mengambil kapasitas stack s.

Functions

- ItemStack newItemStack (int capacity)

  Constructor untuk membuat ItemStack baru.
- boolean isStackEmpty (ItemStack stack)

  Mengecek apakah stack kosong atau tidak.
- boolean isStackFull (ItemStack stack)

  Mengecek apakah stack penuh atau tidak.
- void push (ItemStack *stack, Item item)

  Memasukkan item ke atas stack.
- void pop (ItemStack *stack, Item *item)

  Mengambil item dari atas stack.
- void incrementCapacity (ItemStack *stack)

  Menambah kapasitas stack sebanyak 1. Kapasitas tidak bertambah jika telah mencapai kapasitas maksimum (100).
- void doubleCapacity (ItemStack *stack)

  Menggandakan kapasitas stack. Kapasitas tidak bertambah hingga melebihi kapasitas maksimum (100).
- void _clampCapacity (ItemStack *stack)

  Membatasi kapasitas stack jika kapasitas melebihi batas.

5.33.1 Detailed Description

Header file untuk tipe data ItemStack.

Definition in file item_stack.h.

5.33.2 Macro Definition Documentation

5.33.2.1 capacity

```
#define capacity(
              s ) (s).capacity
```

Mengambil kapasitas stack s.

Parameters

s	ItemStack instance.

Definition at line 50 of file item_stack.h.

5.33.2.2 ITEM_STACK_MAX_CAPACITY

```
#define ITEM_STACK_MAX_CAPACITY 100
```

Kapasitas maksimum ItemStack.

Definition at line 35 of file item_stack.h.

5.33.2.3 top

```
#define top(
              s ) (s).buffer[(s).topIndex]
```

Mengambil Item teratas pada stack s.

Parameters

s	ItemStack instance.

Definition at line 45 of file item_stack.h.

5.33.2.4 topIndex

```
#define topIndex(
              s ) (s).topIndex
```

Mengambil indeks teratas stack s.

Parameters

s	ItemStack instance.

Definition at line 40 of file item_stack.h.

5.33.3 Function Documentation

5.33.3.1 _clampCapacity()

```
void _clampCapacity (
             ItemStack * stack )
```

Membatasi kapasitas stack jika kapasitas melebihi batas.

Parameters

stack	ItemStack instance.

Definition at line 103 of file item_stack.c.

5.33.3.2 doubleCapacity()

```
void doubleCapacity (
             ItemStack * stack )
```

Menggandakan kapasitas stack. Kapasitas tidak bertambah hingga melebihi kapasitas maksimum (100).

Parameters

stack	ItemStack instance.

Definition at line 91 of file item_stack.c.

5.33.3.3 incrementCapacity()

```
void incrementCapacity (
            ItemStack * stack )
```

Menambah kapasitas stack sebanyak 1. Kapasitas tidak bertambah jika telah mencapai kapasitas maksimum (100).

Parameters

stack	ItemStack instance.

Definition at line 78 of file item_stack.c.

5.33.3.4 isStackEmpty()

```
boolean isStackEmpty (
            ItemStack stack )
```

Mengecek apakah stack kosong atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack kosong, false selainnya.

Definition at line 31 of file item_stack.c.

5.33.3.5 isStackFull()

```
boolean isStackFull (
            ItemStack stack )
```

Mengecek apakah stack penuh atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack penuh, false selainnya.

Definition at line 42 of file item_stack.c.

5.33.3.6 newItemStack()

```
ItemStack newItemStack (
            int capacity )
```

Constructor untuk membuat ItemStack baru.

Parameters

capacity	Kapasitas stack.

Returns

ItemStack instace baru yang kosong.

Definition at line 17 of file item_stack.c.

5.33.3.7 pop()

```
void pop (
            ItemStack * stack,
            Item * item )
```

Mengambil item dari atas stack.

Parameters

	stack	ItemStack instance.
out	*item*	Item yang diambil dari atas stack.

Definition at line 65 of file item_stack.c.

5.33.3.8 push()

```
void push (
            ItemStack * stack,
            Item item )
```

Memasukkan item ke atas stack.

Parameters

stack	ItemStack instance.
item	Item yang akan dimasukkan ke atas stack.

Definition at line 53 of file item_stack.c.

5.34 item_stack.h

Go to the documentation of this file.

```
00001
00006 #ifndef ITEM_STACK_H
00007 #define ITEM_STACK_H
00008
00009 #include "boolean.h"
00010 #include "item.h"
00011
00016 typedef struct
00017 {
00021     int topIndex;
00025     int capacity;
00029     Item buffer[100];
00030 } ItemStack;
00031
00035 #define ITEM_STACK_MAX_CAPACITY 100
00040 #define topIndex(s) (s).topIndex
00045 #define top(s) (s).buffer[(s).topIndex]
00050 #define capacity(s) (s).capacity
00051
00058 ItemStack newItemStack(int capacity);
00059
00066 boolean isStackEmpty(ItemStack stack);
00067
00074 boolean isStackFull(ItemStack stack);
00075
00082 void push(ItemStack *stack, Item item);
00083
00090 void pop(ItemStack *stack, Item *item);
00091
00099 void incrementCapacity(ItemStack *stack);
00100
00108 void doubleCapacity(ItemStack *stack);
00109
00116 void _clampCapacity(ItemStack *stack);
00117
00118 #endif
```

5.35 src/models/location.c File Reference

Implementasi tipe data Location. Digunakan untuk merepresentasikan lokasi pada GameMap.

```
#include <stdio.h>
#include "boolean.h"
#include "point.h"
#include "location.h"
#include "../modules/colorizer/colorizer.h"
```

Functions

- Location newLocation (int id, char symbol, Point coordinate)

  Constructor untuk membuat Location baru.
- boolean isAt (Location l, Point p)

  Mengecek apakah suatu lokasi berada pada koordinat (titik) tertentu.
- boolean isLocationIdentical (Location l1, Location l2)

  Mengecek apakah dua lokasi adalah sama atau tidak.
- boolean isLocationDefined (Location l)

  Mengecek apakah suatu lokasi terdefinisi atau tidak.
- void writeLocationSymbol (Location l)

  Menuliskan simbol lokasi ke console output dengan formatting (warna) yang sesuai.
- void setAsPickUpPlace (Location *l)

  Set lokasi sebagai tempat pick up Item.
- void unsetAsPickUpPlace (Location *l)

  Unset lokasi sebagai tempat pick up Item.
- void setAsDropOffPlace (Location *l)

  Set lokasi sebagai tempat drop off Item.
- void unsetAsDropOffPlace (Location *l)

  Unset lokasi sebagai tempat drop off Item.
- void setAsReachable (Location *l)

  Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void unsetAsReachable (Location *l)

  Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void setAsPlayerPlace (Location *l)

  Set lokasi sebagai lokasi player.
- void unsetAsPlayerPlace (Location *l)

  Unset lokasi sebagai lokasi player.
- void toggleAsPlayerPlace (Location *l)

  Toggle lokasi sebagai lokasi player.

Variables

- const Location NULL_LOCATION = {-1, '\0', {-1, -1}, false, false, false}

  Location yang tidak terdefinisi.

5.35.1 Detailed Description

Implementasi tipe data Location. Digunakan untuk merepresentasikan lokasi pada GameMap.

See also

GameMap

Definition in file location.c.

5.35.2 Function Documentation

5.35.2.1 isAt()

```
boolean isAt (
            Location l,
            Point p )
```

Mengecek apakah suatu lokasi berada pada koordinat (titik) tertentu.

Parameters

l	Location instance.
p	Koordinat yang akan dicek (Point instance).

Returns

true jika l berada di p, false selainnya.

Definition at line 49 of file location.c.

5.35.2.2 isLocationDefined()

```
boolean isLocationDefined (
            Location l )
```

Mengecek apakah suatu lokasi terdefinisi atau tidak.

Parameters

l	Location instance.

Returns

true jika l terdefinisi, false selainnya.

Definition at line 74 of file location.c.

5.35.2.3 isLocationIdentical()

```
boolean isLocationIdentical (
            Location l1,
            Location l2 )
```

Mengecek apakah dua lokasi adalah sama atau tidak.

Parameters

l1	Location instance.
l2	Location instance.

Returns

true jika kedua lokasi sama, false selainnya.

Definition at line 62 of file location.c.

5.35.2.4 newLocation()

```
Location newLocation (
            int id,
            char symbol,
            Point coordinate )
```

Constructor untuk membuat Location baru.

Parameters

id	Identifier lokasi.
symbol	Simbol lokasi.
coordinate	Koordinat lokasi.

Returns

Location instance baru.

Definition at line 28 of file location.c.

5.35.2.5 setAsDropOffPlace()

```
void setAsDropOffPlace (
            Location * l )
```

Set lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

Definition at line 139 of file location.c.

5.35.2.6 setAsPickUpPlace()

```
void setAsPickUpPlace (
            Location * l )
```

Set lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

Definition at line 119 of file location.c.

5.35.2.7 setAsPlayerPlace()

```
void setAsPlayerPlace (
            Location * l )
```

Set lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 181 of file location.c.

5.35.2.8 setAsReachable()

```
void setAsReachable (
            Location * l )
```

Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.

Parameters

l	Location instance.

Definition at line 160 of file location.c.

5.35.2.9 toggleAsPlayerPlace()

```
void toggleAsPlayerPlace (
            Location * l )
```

Toggle lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 201 of file location.c.

5.35.2.10 unsetAsDropOffPlace()

```
void unsetAsDropOffPlace (
            Location * l )
```

Unset lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

Definition at line 149 of file location.c.

5.35.2.11 unsetAsPickUpPlace()

```
void unsetAsPickUpPlace (
            Location * l )
```

Unset lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

Definition at line 129 of file location.c.

5.35.2.12 unsetAsPlayerPlace()

```
void unsetAsPlayerPlace (
            Location * l )
```

Unset lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 191 of file location.c.

5.35.2.13 unsetAsReachable()

```
void unsetAsReachable (
            Location * l )
```

Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.

Parameters

l	Location instance.

Definition at line 171 of file location.c.

5.35.2.14 writeLocationSymbol()

```
void writeLocationSymbol (
            Location l )
```

Menuliskan simbol lokasi ke console output dengan formatting (warna) yang sesuai.

Parameters

l	Location instance.

Definition at line 85 of file location.c.

5.35.3 Variable Documentation

5.35.3.1 NULL_LOCATION

```
const Location NULL_LOCATION = {-1, '\0', {-1, -1}, false, false, false}
```

Location yang tidak terdefinisi.

Definition at line 18 of file location.c.

5.36 location.c

Go to the documentation of this file.

```
00001
00009 #include <stdio.h>
00010 #include "boolean.h"
00011 #include "point.h"
00012 #include "location.h"
00013 #include "../modules/colorizer/colorizer.h"
00014
00018 const Location NULL_LOCATION = {-1, '\0', {-1, -1}, false, false, false};
00019
00028 Location newLocation(int id, char symbol, Point coordinate)
00029 {
00030     Location l;
00031     id(l) = id;
00032     symbol(l) = symbol;
00033     coord(l) = coordinate;
00034     unsetAsDropOffPlace(&l);
00035     unsetAsPickUpPlace(&l);
00036     unsetAsPlayerPlace(&l);
00037     unsetAsReachable(&l);
00038     return l;
00039 }
00040
00049 boolean isAt(Location l, Point p)
00050 {
00051     return isPointIdentical(coord(l), p);
00052 }
00053
00062 boolean isLocationIdentical(Location l1, Location l2)
00063 {
00064     return symbol(l1) == symbol(l2) && isPointIdentical(coord(l1), coord(l2)) && id(l1) == id(l2);
00065 }
00066
00074 boolean isLocationDefined(Location l)
00075 {
00076     return !isLocationIdentical(l, NULL_LOCATION);
00077 }
00078
00085 void writeLocationSymbol(Location l)
00086 {
00087     if (isLocationDefined(l))
00088     {
00089         if (l.isPlayerPlace)
00090         {
00091             changeToOrangeColor();
00092         }
00093         else if (l.isDropOffPlace)
00094         {
00095             changeToBlueColor();
00096         }
00097         else if (l.isPickUpPlace)
00098         {
00099             changeToRedColor();
00100         }
00101         else if (l.isReachable)
00102         {
00103             changeToGreenColor();
00104         }
00105         printf(symbol(l));
00106         resetColor();
00107     }
00108     else
00109     {
00110         printf(" ");
00111     }
00112 }
00113
00119 void setAsPickUpPlace(Location *l)
00120 {
00121     l->isPickUpPlace = true;
00122 }
00123
00129 void unsetAsPickUpPlace(Location *l)
00130 {
00131     l->isPickUpPlace = false;
00132 }
00133
00139 void setAsDropOffPlace(Location *l)
00140 {
00141     l->isDropOffPlace = true;
00142 }
00143
00149 void unsetAsDropOffPlace(Location *l)
```

```
00150 {
00151     l->isDropOffPlace = false;
00152 }
00153
00160 void setAsReachable(Location *l)
00161 {
00162     l->isReachable = true;
00163 }
00164
00171 void unsetAsReachable(Location *l)
00172 {
00173     l->isReachable = false;
00174 }
00175
00181 void setAsPlayerPlace(Location *l)
00182 {
00183     l->isPlayerPlace = true;
00184 }
00185
00191 void unsetAsPlayerPlace(Location *l)
00192 {
00193     l->isPlayerPlace = false;
00194 }
00195
00201 void toggleAsPlayerPlace(Location *l)
00202 {
00203     l->isPlayerPlace = !(l->isPlayerPlace);
00204 }
```

5.37 src/models/location.h File Reference

Header file untuk tipe data Location.

```
#include "boolean.h"
#include "point.h"
```

Data Structures

- struct Location

  Struktur tipe data lokasi yang memuat koordinat, simbol, dan id.

Macros

- #define id(l) (l).id

  Mengambil id lokasi.
- #define symbol(l) (l).symbol

  Mengambil simbol lokasi.
- #define coord(l) (l).coordinate

  Mengambil koordinat lokasi.

Functions

- Location newLocation (int id, char symbol, Point coordinate)

  Constructor untuk membuat Location baru.
- boolean isAt (Location l, Point p)

  Mengecek apakah suatu lokasi berada pada koordinat (titik) tertentu.
- boolean isLocationIdentical (Location l1, Location l2)

  Mengecek apakah dua lokasi adalah sama atau tidak.

- boolean isLocationDefined (Location l)

  Mengecek apakah suatu lokasi terdefinisi atau tidak.
- void writeLocationSymbol (Location l)

  Menuliskan simbol lokasi ke console output dengan formatting (warna) yang sesuai.
- void setAsPickUpPlace (Location *l)

  Set lokasi sebagai tempat pick up Item.
- void unsetAsPickUpPlace (Location *l)

  Unset lokasi sebagai tempat pick up Item.
- void setAsDropOffPlace (Location *l)

  Set lokasi sebagai tempat drop off Item.
- void unsetAsDropOffPlace (Location *l)

  Unset lokasi sebagai tempat drop off Item.
- void setAsReachable (Location *l)

  Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void unsetAsReachable (Location *l)

  Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void setAsPlayerPlace (Location *l)

  Set lokasi sebagai lokasi player.
- void unsetAsPlayerPlace (Location *l)

  Unset lokasi sebagai lokasi player.
- void toggleAsPlayerPlace (Location *l)

  Toggle lokasi sebagai lokasi player.

Variables

- Location NULL_LOCATION

  Location yang tidak terdefinisi.

5.37.1 Detailed Description

Header file untuk tipe data Location.

Definition in file location.h.

5.37.2 Macro Definition Documentation

5.37.2.1 coord

```
#define coord(
              l ) (l).coordinate
```

Mengambil koordinat lokasi.

Parameters

l	Location instance.

Definition at line 73 of file location.h.

5.37.2.2 id

```
#define id(
              l ) (l).id
```

Mengambil id lokasi.

Parameters

l	Location instance.

Definition at line 63 of file location.h.

5.37.2.3 symbol

```
#define symbol(
              l ) (l).symbol
```

Mengambil simbol lokasi.

Parameters

l	Location instance.

Definition at line 68 of file location.h.

5.37.3 Function Documentation

5.37.3.1 isAt()

```
boolean isAt (
             Location l,
             Point p )
```

Mengecek apakah suatu lokasi berada pada koordinat (titik) tertentu.

Parameters

l	Location instance.
p	Koordinat yang akan dicek (Point instance).

Returns

true jika l berada di p, false selainnya.

Definition at line 49 of file location.c.

5.37.3.2 isLocationDefined()

```
boolean isLocationDefined (
            Location l )
```

Mengecek apakah suatu lokasi terdefinisi atau tidak.

Parameters

l	Location instance.

Returns

true jika l terdefinisi, false selainnya.

Definition at line 74 of file location.c.

5.37.3.3 isLocationIdentical()

```
boolean isLocationIdentical (
            Location l1,
            Location l2 )
```

Mengecek apakah dua lokasi adalah sama atau tidak.

Parameters

l1	Location instance.
l2	Location instance.

Returns

true jika kedua lokasi sama, false selainnya.

Definition at line 62 of file location.c.

5.37.3.4 newLocation()

```
Location newLocation (
            int id,
            char symbol,
            Point coordinate )
```

Constructor untuk membuat Location baru.

Parameters

id	Identifier lokasi.
symbol	Simbol lokasi.
coordinate	Koordinat lokasi.

Returns

Location instance baru.

Definition at line 28 of file location.c.

5.37.3.5 setAsDropOffPlace()

```
void setAsDropOffPlace (
            Location * l )
```

Set lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

Definition at line 139 of file location.c.

5.37.3.6 setAsPickUpPlace()

```
void setAsPickUpPlace (
            Location * l )
```

Set lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

Definition at line 119 of file location.c.

5.37.3.7 setAsPlayerPlace()

```
void setAsPlayerPlace (
            Location * l )
```

Set lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 181 of file location.c.

5.37.3.8 setAsReachable()

```
void setAsReachable (
            Location * l )
```

Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.

Parameters

l	Location instance.

Definition at line 160 of file location.c.

5.37.3.9 toggleAsPlayerPlace()

```
void toggleAsPlayerPlace (
            Location * l )
```

Toggle lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 201 of file location.c.

5.37.3.10 unsetAsDropOffPlace()

```
void unsetAsDropOffPlace (
            Location * l )
```

Unset lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

Definition at line 149 of file location.c.

5.37.3.11 unsetAsPickUpPlace()

```
void unsetAsPickUpPlace (
            Location * l )
```

Unset lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

Definition at line 129 of file location.c.

5.37.3.12 unsetAsPlayerPlace()

```
void unsetAsPlayerPlace (
            Location * l )
```

Unset lokasi sebagai lokasi player.

Parameters

l	Location instance.

Definition at line 191 of file location.c.

5.37.3.13 unsetAsReachable()

```
void unsetAsReachable (
            Location * l )
```

Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.

Parameters

l	Location instance.

Definition at line 171 of file location.c.

5.37.3.14 writeLocationSymbol()

```
void writeLocationSymbol (
            Location l )
```

Menuliskan simbol lokasi ke console output dengan formatting (warna) yang sesuai.

Parameters

l	Location instance.

Definition at line 85 of file location.c.

5.37.4 Variable Documentation

5.37.4.1 NULL_LOCATION

```
Location NULL_LOCATION  [extern]
```

Location yang tidak terdefinisi.

Definition at line 18 of file location.c.

5.38 location.h

Go to the documentation of this file.

```
00001
00006 #ifndef LOCATION_H
00007 #define LOCATION_H
00008
00009 #include "boolean.h"
00010 #include "point.h"
```

```
00011
00015 extern Location NULL_LOCATION;
00016
00022 typedef struct
00023 {
00028     int id;
00032     char symbol;
00036     Point coordinate;
00041     boolean isPlayerPlace;
00046     boolean isPickUpPlace;
00051     boolean isDropOffPlace;
00056     boolean isReachable;
00057 } Location;
00058
00063 #define id(l) (l).id
00068 #define symbol(l) (l).symbol
00073 #define coord(l) (l).coordinate
00074
00083 Location newLocation(int id, char symbol, Point coordinate);
00084
00093 boolean isAt(Location l, Point p);
00094
00103 boolean isLocationIdentical(Location l1, Location l2);
00104
00112 boolean isLocationDefined(Location l);
00113
00120 void writeLocationSymbol(Location l);
00121
00127 void setAsPickUpPlace(Location *l);
00128
00134 void unsetAsPickUpPlace(Location *l);
00135
00141 void setAsDropOffPlace(Location *l);
00142
00148 void unsetAsDropOffPlace(Location *l);
00149
00156 void setAsReachable(Location *l);
00157
00164 void unsetAsReachable(Location *l);
00165
00171 void setAsPlayerPlace(Location *l);
00172
00178 void unsetAsPlayerPlace(Location *l);
00179
00185 void toggleAsPlayerPlace(Location *l);
00186
00187 #endif
```

5.39 src/models/location_list.c File Reference

Implementasi tipe data LocationList. Digunakan menyimpan daftar lokasi yang ada, dan daftar lokasi yang adjacent dengan suatu lokasi.

```
#include <stdlib.h>
#include "boolean.h"
#include "location.h"
#include "point.h"
#include "location_list.h"
```

Functions

- LocationList newLocationList (int capacity)

  Constructor untuk membuat LocationList baru.
- void dealocateLocationList (LocationList *l)

  Menghapus memory list dari heap memory.
- int length (LocationList l)

  Mengembalikan panjang list l.
- boolean isIndexValid (LocationList l, int i)

Mengecek apakah suatu indeks i adalah indeks yang valid untuk list l.

- boolean isIndexEff (LocationList l, int i)

  Mengecek apakah suatu indeks i adalah indeks yang efektif untuk list l.
- boolean isLocationListEmpty (LocationList l)

  Mengecek apakah list l kosong atau tidak.
- boolean isLocationListFull (LocationList l)

  Mengecek apakah list l penuh atau tidak.
- void insertLast (LocationList *l, Location location)

  Memasukkan Location ke akhir list.
- void deleteLast (LocationList *l, Location *val)

  Menghapus & mengambil Location terakhir di dalam list.
- void growList (LocationList *l, int num)

  Menambah kapasitas list l sebanyak num.
- void shrinkList (LocationList *l, int num)

  Mengurangi kapasitas list l sebanyak num.
- void compactList (LocationList *l)

  Merapatkan list l.
- void sortLocationListByCoord (LocationList *l)

  Mengurutkan Location dalam l berdasarkan koordinat.
- Location _getLocationById (LocationList l, int id)

  Mengambil lokasi dalam list l berdasarkan id lokasi.
- Location _getLocationBySymbol (LocationList l, char symbol)

  Mengambil lokasi dalam list l berdasarkan simbol lokasi.
- Location _getLocationByCoord (LocationList l, Point p)

  Mengambil lokasi dalam list l berdasarkan koordinat lokasi.

5.39.1 Detailed Description

Implementasi tipe data LocationList. Digunakan menyimpan daftar lokasi yang ada, dan daftar lokasi yang adjacent dengan suatu lokasi.

Definition in file location_list.c.

5.39.2 Function Documentation

5.39.2.1 _getLocationByCoord()

```
Location _getLocationByCoord (
            LocationList l,
            Point p )
```

Mengambil lokasi dalam list l berdasarkan koordinat lokasi.

Parameters

l	LocationList instance.
p	Koordinat lokasi yang akan diambil.

Returns

Lokasi dengan koordinat p, atau NULL_LOCATION jika lokasi dengan koordinat p tidak ditemukan dalam l.

Definition at line 246 of file location_list.c.

5.39.2.2 _getLocationById()

```
Location _getLocationById (
            LocationList l,
            int id )
```

Mengambil lokasi dalam list l berdasarkan id lokasi.

Parameters

l	LocationList instance.
id	Id lokasi yang akan diambil.

Returns

Lokasi dengan id 'id', atau NULL_LOCATION jika lokasi dengan id 'id' tidak ditemukan dalam l.

Definition at line 198 of file location_list.c.

5.39.2.3 _getLocationBySymbol()

```
Location _getLocationBySymbol (
            LocationList l,
            char symbol )
```

Mengambil lokasi dalam list l berdasarkan simbol lokasi.

Parameters

l	LocationList instance.
symbol	Simbol lokasi yang akan diambil.

Returns

Lokasi dengan simbol symbol, atau NULL_LOCATION jika lokasi dengan simbol symbol tidak ditemukan dalam l.

Definition at line 222 of file location_list.c.

5.39.2.4 compactList()

```
void compactList (
            LocationList * l )
```

Merapatkan list l.

Parameters

l	LocationList instance.

Definition at line 159 of file location_list.c.

5.39.2.5 dealocateLocationList()

```
void dealocateLocationList (
            LocationList * l )
```

Menghapus memory list dari heap memory.

Parameters

l	LocationList instance.

Definition at line 35 of file location_list.c.

5.39.2.6 deleteLast()

```
void deleteLast (
            LocationList * l,
            Location * val )
```

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	l	LocaitonList instance.
out	*location*	Location instance.

Definition at line 120 of file location_list.c.

5.39.2.7 growList()

```
void growList (
```

```
            LocationList * l,
            int num )
```

Menambah kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan ditambah.

Definition at line 132 of file location_list.c.

5.39.2.8 insertLast()

```
void insertLast (
            LocationList * l,
            Location location )
```

Memasukkan Location ke akhir list.

Parameters

l	LocationList instance.
location	Location instance.

Definition at line 107 of file location_list.c.

5.39.2.9 isIndexEff()

```
boolean isIndexEff (
            LocationList l,
            int i )
```

Mengecek apakah suatu indeks i adalah indeks yang efektif untuk list l.

Parameters

l	LocationList instance.
i	Indeks yang akan dicek.

Returns

true jika indeks efektif, false selainnya.

Definition at line 74 of file location_list.c.

5.39.2.10 isIndexValid()

```
boolean isIndexValid (
            LocationList l,
            int i )
```

Mengecek apakah suatu indeks i adalah indeks yang valid untuk list l.

Parameters

l	LocationList instance.
i	Indeks yang akan dicek.

Returns

true jika indeks valid, false selainnya.

Definition at line 61 of file location_list.c.

5.39.2.11 isLocationListEmpty()

```
boolean isLocationListEmpty (
            LocationList l )
```

Mengecek apakah list l kosong atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l kosong, false selainnya.

Definition at line 85 of file location_list.c.

5.39.2.12 isLocationListFull()

```
boolean isLocationListFull (
            LocationList l )
```

Mengecek apakah list l penuh atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l penuh, false selainnya.

Definition at line 96 of file location_list.c.

5.39.2.13 length()

```
int length (
            LocationList l )
```

Mengembalikan panjang list l.

Parameters

l	LocationList instance.

Returns

Panjang list l.

Definition at line 48 of file location_list.c.

5.39.2.14 newLocationList()

```
LocationList newLocationList (
            int capacity )
```

Constructor untuk membuat LocationList baru.

Parameters

capacity	Kapasitas list.

Returns

LocationList instance baru yang kosong.

Definition at line 21 of file location_list.c.

5.39.2.15 shrinkList()

```
void shrinkList (
            LocationList * l,
            int num )
```

Mengurangi kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan dikurangi.

Definition at line 144 of file location_list.c.

5.39.2.16 sortLocationListByCoord()

```
void sortLocationListByCoord (
            LocationList * l )
```

Mengurutkan Location dalam l berdasarkan koordinat.

See also

Location

Parameters

l	LocationList instance.

Definition at line 171 of file location_list.c.

5.40 location_list.c

Go to the documentation of this file.

```
00001
00009 #include <stdlib.h>
00010 #include "boolean.h"
00011 #include "location.h"
00012 #include "point.h"
00013 #include "location_list.h"
00014
00021 LocationList newLocationList(int capacity)
00022 {
00023     LocationList l;
00024     capacity(l) = capacity;
00025     buffer(l) = (Location *)malloc(capacity * sizeof(Location));
00026     neff(l) = 0;
00027     return l;
00028 }
00029
00035 void dealocateLocationList(LocationList *l)
00036 {
00037     free(buffer(*l));
00038     neff(*l) = 0;
00039     capacity(*l) = 0;
00040 }
00041
00048 int length(LocationList l)
00049 {
00050     return neff(l);
00051 }
00052
00061 boolean isIndexValid(LocationList l, int i)
00062 {
00063     return i < capacity(l);
```

```
00064 }
00065
00074 boolean isIndexEff(LocationList l, int i)
00075 {
00076     return i < length(l);
00077 }
00078
00085 boolean isLocationListEmpty(LocationList l)
00086 {
00087     return length(l) == 0;
00088 }
00089
00096 boolean isLocationListFull(LocationList l)
00097 {
00098     return length(l) == capacity(l);
00099 }
00100
00107 void insertLast(LocationList *l, Location location)
00108 {
00109     lElem(*l, length(*l)) = location;
00110     neff(*l)++;
00111 }
00112
00120 void deleteLast(LocationList *l, Location *val)
00121 {
00122     neff(*l)--;
00123     *val = lElem(*l, length(*l));
00124 }
00125
00132 void growList(LocationList *l, int num)
00133 {
00134     capacity(*l) += num;
00135     buffer(*l) = (Location *)realloc(buffer(*l), capacity(*l) * sizeof(Location));
00136 }
00137
00144 void shrinkList(LocationList *l, int num)
00145 {
00146     capacity(*l) -= num;
00147     if (neff(*l) > capacity(*l))
00148     {
00149         neff(*l) = capacity(*l);
00150     }
00151     buffer(*l) = (Location *)realloc(buffer(*l), capacity(*l) * sizeof(Location));
00152 }
00153
00159 void compactList(LocationList *l)
00160 {
00161     const int diff = capacity(*l) - neff(*l);
00162     shrinkList(l, diff);
00163 }
00164
00171 void sortLocationListByCoord(LocationList *l)
00172 {
00173     for (int i = 0; i < length(*l) - 1; i++)
00174     {
00175         int priorityIndex = i;
00176         for (int j = i + 1; j < length(*l); j++)
00177         {
00178             if (isPointBefore(coord(lElem(*l, j)), coord(lElem(*l, priorityIndex))))
00179             {
00180                 priorityIndex = j;
00181             }
00182         }
00183         Location temp = lElem(*l, i);
00184         lElem(*l, i) = lElem(*l, priorityIndex);
00185         lElem(*l, priorityIndex) = temp;
00186     }
00187 }
00188
00198 Location _getLocationById(LocationList l, int id)
00199 {
00200     Location loc;
00201     int i = 0;
00202     while (i < length(l))
00203     {
00204         loc = lElem(l, i);
00205         if (id(loc) == id)
00206         {
00207             return loc;
00208         }
00209     }
00210     return NULL_LOCATION;
00211 }
00212
00222 Location _getLocationBySymbol(LocationList l, char symbol)
00223 {
00224     Location loc;
```

```
00225     int i = 0;
00226     while (i < length(l))
00227     {
00228         loc = lElem(l, i);
00229         if (symbol(loc) == symbol)
00230         {
00231             return loc;
00232         }
00233     }
00234     return NULL_LOCATION;
00235 }
00236
00246 Location _getLocationByCoord(LocationList l, Point p)
00247 {
00248     Location loc;
00249     int i = 0;
00250     while (i < length(l))
00251     {
00252         loc = lElem(l, i);
00253         if (isPointIdentical(coord(loc), p))
00254         {
00255             return loc;
00256         }
00257     }
00258     return NULL_LOCATION;
00259 }
```

5.41 src/models/location_list.h File Reference

Header file untuk tipe data LocationList.

```
#include "boolean.h"
#include "location.h"
#include "point.h"
```

Data Structures

- struct LocationList

  List dinamis berisi data Location.

Macros

- #define neff(l) (l).nEff

  Mengambil banyak elemen list.
- #define capacity(l) (l).capacity

  Mengambil kapasitas elemen list.
- #define buffer(l) (l).buffer

  Mengambil memory list.
- #define lElem(l, i) (l).buffer[i]

  Mengambil elemen list pada indeks tertentu.

Functions

- LocationList newLocationList (int capacity)

  Constructor untuk membuat LocationList baru.
- void dealocateLocationList (LocationList *l)

  Menghapus memory list dari heap memory.
- int length (LocationList l)

  Mengembalikan panjang list l.
- boolean isIndexValid (LocationList l, int i)

  Mengecek apakah suatu indeks i adalah indeks yang valid untuk list l.
- boolean isIndexEff (LocationList l, int i)

  Mengecek apakah suatu indeks i adalah indeks yang efektif untuk list l.
- boolean isLocationListEmpty (LocationList l)

  Mengecek apakah list l kosong atau tidak.
- boolean isLocationListFull (LocationList l)

  Mengecek apakah list l penuh atau tidak.
- void insertLast (LocationList *l, Location location)

  Memasukkan Location ke akhir list.
- void deleteLast (LocationList *l, Location *location)

  Menghapus & mengambil Location terakhir di dalam list.
- void growList (LocationList *l, int num)

  Menambah kapasitas list l sebanyak num.
- void shrinkList (LocationList *l, int num)

  Mengurangi kapasitas list l sebanyak num.
- void compactList (LocationList *l)

  Merapatkan list l.
- void sortLocationListByCoord (LocationList *l)

  Mengurutkan Location dalam l berdasarkan koordinat.
- Location _getLocationById (LocationList l, int id)

  Mengambil lokasi dalam list l berdasarkan id lokasi.
- Location _getLocationBySymbol (LocationList l, char symbol)

  Mengambil lokasi dalam list l berdasarkan simbol lokasi.
- Location _getLocationByCoord (LocationList l, Point p)

  Mengambil lokasi dalam list l berdasarkan koordinat lokasi.

5.41.1 Detailed Description

Header file untuk tipe data LocationList.

Definition in file location_list.h.

5.41.2 Macro Definition Documentation

5.41.2.1 buffer

```
#define buffer(
               l ) (l).buffer
```

Mengambil memory list.

Parameters

l	LocationList instance.

Definition at line 47 of file location_list.h.

5.41.2.2 capacity

```
#define capacity(
              l ) (l).capacity
```

Mengambil kapasitas elemen list.

Parameters

l	LocationList instance.

Definition at line 42 of file location_list.h.

5.41.2.3 lElem

```
#define lElem(
              l,
              i ) (l).buffer[i]
```

Mengambil elemen list pada indeks tertentu.

Parameters

l	LocationList instance.
i	Indeks elemen yang akan diambil.

Definition at line 53 of file location_list.h.

5.41.2.4 neff

```
#define neff(
              l ) (l).nEff
```

Mengambil banyak elemen list.

Parameters

l	LocationList instance.

Definition at line 37 of file location_list.h.

5.41.3 Function Documentation

5.41.3.1 _getLocationByCoord()

```
Location _getLocationByCoord (
            LocationList l,
            Point p )
```

Mengambil lokasi dalam list l berdasarkan koordinat lokasi.

Parameters

l	LocationList instance.
p	Koordinat lokasi yang akan diambil.

Returns

Lokasi dengan koordinat p, atau NULL_LOCATION jika lokasi dengan koordinat p tidak ditemukan dalam l.

Definition at line 246 of file location_list.c.

5.41.3.2 _getLocationById()

```
Location _getLocationById (
            LocationList l,
            int id )
```

Mengambil lokasi dalam list l berdasarkan id lokasi.

Parameters

l	LocationList instance.
id	Id lokasi yang akan diambil.

Returns

Lokasi dengan id 'id', atau NULL_LOCATION jika lokasi dengan id 'id' tidak ditemukan dalam l.

Definition at line 198 of file location_list.c.

5.41.3.3 _getLocationBySymbol()

```
Location _getLocationBySymbol (
            LocationList l,
            char symbol )
```

Mengambil lokasi dalam list l berdasarkan simbol lokasi.

Parameters

l	LocationList instance.
symbol	Simbol lokasi yang akan diambil.

Returns

Lokasi dengan simbol symbol, atau NULL_LOCATION jika lokasi dengan simbol symbol tidak ditemukan dalam l.

Definition at line 222 of file location_list.c.

5.41.3.4 compactList()

```
void compactList (
            LocationList * l )
```

Merapatkan list l.

Parameters

l	LocationList instance.

Definition at line 159 of file location_list.c.

5.41.3.5 dealocateLocationList()

```
void dealocateLocationList (
            LocationList * l )
```

Menghapus memory list dari heap memory.

Parameters

l	LocationList instance.

Definition at line 35 of file location_list.c.

5.41.3.6 deleteLast()

```
void deleteLast (
            LocationList * l,
            Location * val )
```

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	l	LocaitonList instance.
out	*location*	Location instance.

Definition at line 120 of file location_list.c.

5.41.3.7 growList()

```
void growList (
            LocationList * l,
            int num )
```

Menambah kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan ditambah.

Definition at line 132 of file location_list.c.

5.41.3.8 insertLast()

```
void insertLast (
            LocationList * l,
            Location location )
```

Memasukkan Location ke akhir list.

Parameters

l	LocationList instance.
location	Location instance.

Definition at line 107 of file location_list.c.

5.41.3.9 isIndexEff()

```
boolean isIndexEff (
            LocationList l,
            int i )
```

Mengecek apakah suatu indeks i adalah indeks yang efektif untuk list l.

Parameters

l	LocationList instance.
i	Indeks yang akan dicek.

Returns

true jika indeks efektif, false selainnya.

Definition at line 74 of file location_list.c.

5.41.3.10 isIndexValid()

```
boolean isIndexValid (
            LocationList l,
            int i )
```

Mengecek apakah suatu indeks i adalah indeks yang valid untuk list l.

Parameters

l	LocationList instance.
i	Indeks yang akan dicek.

Returns

true jika indeks valid, false selainnya.

Definition at line 61 of file location_list.c.

5.41.3.11 isLocationListEmpty()

```
boolean isLocationListEmpty (
            LocationList l )
```

Mengecek apakah list l kosong atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l kosong, false selainnya.

Definition at line 85 of file location_list.c.

5.41.3.12 isLocationListFull()

```
boolean isLocationListFull (
            LocationList l )
```

Mengecek apakah list l penuh atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l penuh, false selainnya.

Definition at line 96 of file location_list.c.

5.41.3.13 length()

```
int length (
            LocationList l )
```

Mengembalikan panjang list l.

Parameters

l	LocationList instance.

Returns

Panjang list l.

Definition at line 48 of file location_list.c.

5.41.3.14 newLocationList()

```
LocationList newLocationList (
            int capacity )
```

Constructor untuk membuat LocationList baru.

Parameters

capacity	Kapasitas list.

Returns

LocationList instance baru yang kosong.

Definition at line 21 of file location_list.c.

5.41.3.15 shrinkList()

```
void shrinkList (
            LocationList * l,
            int num )
```

Mengurangi kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan dikurangi.

Definition at line 144 of file location_list.c.

5.41.3.16 sortLocationListByCoord()

```
void sortLocationListByCoord (
            LocationList * l )
```

Mengurutkan Location dalam l berdasarkan koordinat.

See also

Location

Parameters

l	LocationList instance.

Definition at line 171 of file location_list.c.

5.42 location_list.h

Go to the documentation of this file.

```
00001
00006 #ifndef LOCATION_LIST_H
00007 #define LOCATION_LIST_H
00008
00009 #include "boolean.h"
00010 #include "location.h"
00011 #include "point.h"
00012
00017 typedef struct
00018 {
00022     Location *buffer;
00026     int nEff;
00030     int capacity;
00031 } LocationList;
00032
00037 #define neff(l) (l).nEff
00042 #define capacity(l) (l).capacity
00047 #define buffer(l) (l).buffer
00053 #define lElem(l, i) (l).buffer[i]
00054
00061 LocationList newLocationList(int capacity);
00062
00068 void dealocateLocationList(LocationList *l);
00069
00076 int length(LocationList l);
00077
00086 boolean isIndexValid(LocationList l, int i);
00087
00096 boolean isIndexEff(LocationList l, int i);
00097
00104 boolean isLocationListEmpty(LocationList l);
00105
00112 boolean isLocationListFull(LocationList l);
00113
00120 void insertLast(LocationList *l, Location location);
00121
00129 void deleteLast(LocationList *l, Location *location);
00130
00137 void growList(LocationList *l, int num);
00138
00145 void shrinkList(LocationList *l, int num);
00146
00152 void compactList(LocationList *l);
00153
00160 void sortLocationListByCoord(LocationList *l);
00161
00171 Location _getLocationById(LocationList l, int id);
00172
00182 Location _getLocationBySymbol(LocationList l, char symbol);
00183
00193 Location _getLocationByCoord(LocationList l, Point p);
00194
00195 #endif
```

5.43 src/models/location_matrix.c File Reference

```
#include "boolean.h"
#include "location.h"
#include "location_matrix.h"
```

Functions

- LocationMatrix newLocationMatrix (int rows, int cols)

  Constructor untuk membuat LocationMatrix baru.
- void lSetElem (LocationMatrix *locationMatrix, int rowIndex, int colIndex, Location location)

  Set elemen LocationMatrix l pada indeks tertentu.

5.43.1 Function Documentation

5.43.1.1 lSetElem()

```
void lSetElem (
            LocationMatrix * locationMatrix,
            int rowIndex,
            int colIndex,
            Location location )
```

Set elemen LocationMatrix l pada indeks tertentu.

Parameters

l	LocationMatrix instance.
rowIndex	Indeks baris yang akan di-set.
colIndex	Indeks kolom yang akan di-set.
value	Location instance.

Definition at line 41 of file location_matrix.c.

5.43.1.2 newLocationMatrix()

```
LocationMatrix newLocationMatrix (
            int rows,
            int cols )
```

Constructor untuk membuat LocationMatrix baru.

Parameters

rows	Banyak baris matriks.
cols	Banyak kolom matriks.

Returns

LocationMatrix instance baru yang kosong.

Definition at line 17 of file location_matrix.c.

5.44 location_matrix.c

Go to the documentation of this file.

```
00001
00006 #include "boolean.h"
00007 #include "location.h"
00008 #include "location_matrix.h"
00009
00017 LocationMatrix newLocationMatrix(int rows, int cols)
00018 {
00019     LocationMatrix locationMatrix;
00020     rows(locationMatrix) = rows;
00021     cols(locationMatrix) = cols;
00022
00023     for (int i = 0; i < rows(locationMatrix); i++)
00024     {
00025         for (int j = 0; j < cols(locationMatrix); j++)
00026         {
00027             elem(locationMatrix, i, j) = NULL_LOCATION;
00028         }
00029     }
00030 }
00031
00041 void lSetElem(LocationMatrix *locationMatrix, int rowIndex, int colIndex, Location location)
00042 {
00043     elem(*locationMatrix, rowIndex, colIndex) = location;
00044 }
```

5.45 src/models/location_matrix.h File Reference

Implementasi tipe data LocationMatrix.

```
#include "boolean.h"
#include "location.h"
```

Data Structures

- struct LocationMatrix

  Matriks berisi data Location.

Macros

- #define rows(l) (l).rowEff

  Mengambil banyak baris matriks l.
- #define cols(l) (l).colEff

  Mengambil banyak kolom matriks l.
- #define elem(l, i, j) (l).contents[i][j]

  Mengambil Location pada matrix l pada indeks (i, j).

Functions

- LocationMatrix newLocationMatrix (int rows, int cols)

  Constructor untuk membuat LocationMatrix baru.
- void lSetElem (LocationMatrix *l, int rowIndex, int colIndex, Location value)

  Set elemen LocationMatrix l pada indeks tertentu.

5.45.1 Detailed Description

Implementasi tipe data LocationMatrix.

Header file untuk tipe data LocationMatrix.

Definition in file location_matrix.h.

5.45.2 Macro Definition Documentation

5.45.2.1 cols

```
#define cols(
             l ) (l).colEff
```

Mengambil banyak kolom matriks l.

Parameters

l	LocationMatrix instance.

Definition at line 41 of file location_matrix.h.

5.45.2.2 elem

```
#define elem(
             l,
             i,
             j ) (l).contents[i][j]
```

Mengambil Location pada matrix l pada indeks (i, j).

Parameters

l	LocationMatrix instance.
i	Indeks baris elemen yang akan diambil.
j	Indeks kolom elemen yang akan diambil.

Definition at line 49 of file location_matrix.h.

5.45.2.3 rows

```
#define rows(
              l ) (l).rowEff
```

Mengambil banyak baris matriks l.

Parameters

l	LocationMatrix instance.

Definition at line 36 of file location_matrix.h.

5.45.3 Function Documentation

5.45.3.1 lSetElem()

```
void lSetElem (
             LocationMatrix * locationMatrix,
             int rowIndex,
             int colIndex,
             Location location )
```

Set elemen LocationMatrix l pada indeks tertentu.

Parameters

l	LocationMatrix instance.
rowIndex	Indeks baris yang akan di-set.
colIndex	Indeks kolom yang akan di-set.
value	Location instance.

Definition at line 41 of file location_matrix.c.

5.45.3.2 newLocationMatrix()

```
LocationMatrix newLocationMatrix (
             int rows,
             int cols )
```

Constructor untuk membuat LocationMatrix baru.

Parameters

rows	Banyak baris matriks.
cols	Banyak kolom matriks.

Returns

LocationMatrix instance baru yang kosong.

Definition at line 17 of file location_matrix.c.

5.46 location_matrix.h

Go to the documentation of this file.

```
00001
00006 #ifndef LOCATION_MATRIX_H
00007 #define LOCATION_MATRIX_H
00008
00009 #include "boolean.h"
00010 #include "location.h"
00011
00016 typedef struct
00017 {
00021     Location contents[20][30];
00025     int rowEff;
00029     int colEff;
00030 } LocationMatrix;
00031
00036 #define rows(l) (l).rowEff
00041 #define cols(l) (l).colEff
00049 #define elem(l, i, j) (l).contents[i][j]
00050
00058 LocationMatrix newLocationMatrix(int rows, int cols);
00059
00069 void lSetElem(LocationMatrix *l, int rowIndex, int colIndex, Location value);
00070
00071 #endif
```

5.47 src/models/point.c File Reference

Implementasi tipe data Point. Digunakan untuk merepresentasikan sebuah koordinat lokasi.

```
#include <stdio.h>
#include "boolean.h"
#include "point.h"
```

Functions

- Point newPoint (int x, int y)

  Constructor untuk membuat Point baru.
- boolean isPointIdentical (Point p1, Point p2)

  Mengecek apakah dua titik sama atau tidak.
- boolean isPointBefore (Point p1, Point p2)

  Mengecek apakah suatu titik berada pada koordinat "sebelum" titik lainnya, yaitu x1 < x2 atau x1 = x2 & y1 < y2.
- void displayPoint (Point p)

  Menuliskan titik p ke console output.

5.47.1 Detailed Description

Implementasi tipe data Point. Digunakan untuk merepresentasikan sebuah koordinat lokasi.

See also

Location

Definition in file point.c.

5.47.2 Function Documentation

5.47.2.1 displayPoint()

```
void displayPoint (
            Point p )
```

Menuliskan titik p ke console output.

Parameters

p	Point instance.

Definition at line 59 of file point.c.

5.47.2.2 isPointBefore()

```
boolean isPointBefore (
            Point p1,
            Point p2 )
```

Mengecek apakah suatu titik berada pada koordinat "sebelum" titik lainnya, yaitu x1 $<$ x2 atau x1 = x2 & y1 $<$ y2.

Parameters

p1	Point instance.
p2	Point instance.

Returns

true jika titik pertama berada pada koordinat "sebelum" titik kedua, false selainnya.

Definition at line 49 of file point.c.

5.47.2.3 isPointIdentical()

```
boolean isPointIdentical (
            Point p1,
            Point p2 )
```

Mengecek apakah dua titik sama atau tidak.

Parameters

p1	Point instance.
p2	Point instance.

Returns

true jika kedua titik sama, false selainnya.

Definition at line 34 of file point.c.

5.47.2.4 newPoint()

```
Point newPoint (
            int x,
            int y )
```

Constructor untuk membuat Point baru.

Parameters

x	Absis.
y	Ordinat.

Returns

Point instance baru dengan koordinat (x, y).

Definition at line 20 of file point.c.

5.48 point.c

Go to the documentation of this file.

```
00001
00009 #include <stdio.h>
00010 #include "boolean.h"
00011 #include "point.h"
00012
00020 Point newPoint(int x, int y)
00021 {
00022     Point p;
```

```
00023     abs(p) = x;
00024     ord(p) = y;
00025 }
00026
00034 boolean isPointIdentical(Point p1, Point p2)
00035 {
00036     return abs(p1) == abs(p2) && ord(p1) == ord(p2);
00037 }
00038
00049 boolean isPointBefore(Point p1, Point p2)
00050 {
00051     return abs(p1) < abs(p2) || (abs(p1) == abs(p2) && ord(p1) < ord(p2));
00052 }
00053
00059 void displayPoint(Point p)
00060 {
00061     printf("(%d, %d)", abs(p), ord(p));
00062 }
```

5.49 src/models/point.h File Reference

Header file untuk tipe data Point.

```
#include "boolean.h"
```

Data Structures

- struct Point

  Struktur tipe data titik.

Macros

- #define abs(p) (p).x

  Mengambil absis dari suatu titik.
- #define ord(p) (p).y

  Mengambil ordinat dari suatu titik.

Functions

- Point newPoint (int x, int y)

  Constructor untuk membuat Point baru.
- boolean isPointIdentical (Point p1, Point p2)

  Mengecek apakah dua titik sama atau tidak.
- boolean isPointBefore (Point p1, Point p2)

  Mengecek apakah suatu titik berada pada koordinat "sebelum" titik lainnya, yaitu x1 < x2 atau x1 = x2 & y1 < y2.
- void displayPoint (Point p)

  Menuliskan titik p ke console output.

5.49.1 Detailed Description

Header file untuk tipe data Point.

Definition in file point.h.

5.49.2 Macro Definition Documentation

5.49.2.1 abs

```
#define abs(
             p ) (p).x
```

Mengambil absis dari suatu titik.

Parameters

p	Point instance.

Definition at line 40 of file point.h.

5.49.2.2 ord

```
#define ord(
             p ) (p).y
```

Mengambil ordinat dari suatu titik.

Parameters

p	Point instance.

Definition at line 45 of file point.h.

5.49.3 Function Documentation

5.49.3.1 displayPoint()

```
void displayPoint (
            Point p )
```

Menuliskan titik p ke console output.

Parameters

p	Point instance.

Definition at line 59 of file point.c.

5.49.3.2 isPointBefore()

```
boolean isPointBefore (
            Point p1,
            Point p2 )
```

Mengecek apakah suatu titik berada pada koordinat "sebelum" titik lainnya, yaitu x1 < x2 atau x1 = x2 & y1 < y2.

Parameters

p1	Point instance.
p2	Point instance.

Returns

true jika titik pertama berada pada koordinat "sebelum" titik kedua, false selainnya.

Definition at line 49 of file point.c.

5.49.3.3 isPointIdentical()

```
boolean isPointIdentical (
            Point p1,
            Point p2 )
```

Mengecek apakah dua titik sama atau tidak.

Parameters

p1	Point instance.
p2	Point instance.

Returns

true jika kedua titik sama, false selainnya.

Definition at line 34 of file point.c.

5.49.3.4 newPoint()

```
Point newPoint (
            int x,
            int y )
```

Constructor untuk membuat Point baru.

Parameters

x	Absis.
y	Ordinat.

Returns

Point instance baru dengan koordinat (x, y).

Definition at line 20 of file point.c.

5.50 point.h

Go to the documentation of this file.

```
00001
00006 #ifndef POINT_H
00007 #define POINT_H
00008
00009 #include "boolean.h"
00010
00015 typedef struct
00016 {
00020     int x;
00024     int y;
00025 } Point;
00026
00034 Point newPoint(int x, int y);
00035
00040 #define abs(p) (p).x
00045 #define ord(p) (p).y
00046
00054 boolean isPointIdentical(Point p1, Point p2);
00055
00066 boolean isPointBefore(Point p1, Point p2);
00067
00073 void displayPoint(Point p);
00074
00075 #endif
```

5.51 src/models/state.h File Reference

Header file untuk State game.

```
#include "game_map.h"
#include "item_list.h"
#include "item_queue.h"
#include "item_stack.h"
#include "gadget_list.h"
```

Data Structures

- struct State

  Game state & life cycle.

Functions

- State newState (GameMap m, ItemList todo, ItemList inProgress, ItemStack bag, ItemQueue order)

  Constructor untuk membuat instance (termasuk State) game yang baru.
- void moveItemToProgressList (State *state, int indexTodo)

  Memindahkan Item dalam Todo List ke In Progress List. Hanya dipanggil ketika player melakukan pick up Item.
- void reevaluate (State *state)

  Reevaluasi state setelah player menjalankan suatu command atau setelah waktu bertambah.

5.51.1 Detailed Description

Header file untuk State game.

Definition in file state.h.

5.51.2 Function Documentation

5.51.2.1 moveItemToProgressList()

```
void moveItemToProgressList (
            State * state,
            int indexTodo )
```

Memindahkan Item dalam Todo List ke In Progress List. Hanya dipanggil ketika player melakukan pick up Item.

Parameters

state	State saat ini.
indexTodo	Indeks item pada todo yang akan dipindahkan.

5.51.2.2 newState()

```
State newState (
            GameMap m,
            ItemList todo,
            ItemList inProgress,
```

```
            ItemStack bag,
            ItemQueue order )
```

Constructor untuk membuat instance (termasuk State) game yang baru.

Todo Implementasi State, termasuk fungsi-fungsi yang mengubah State game, save State, dan reevaluasi State setiap player menjalankan command.

Parameters

m	GameMap instance dari game instance ini.
todo	Todo List dari game instance ini.
inProgress	In Progress List dari game instance ini.
bag	Tas player pada game instance ini.
order	Daftar pesanan pada game instance ini.

Returns

State

5.51.2.3 reevaluate()

```
void reevaluate (
            State * state )
```

Reevaluasi state setelah player menjalankan suatu command atau setelah waktu bertambah.

Parameters

state	State saat ini.

5.52 state.h

Go to the documentation of this file.

```
00001
00006 #ifndef STATE_H
00007 #define STATE_H
00008
00009 #include "game_map.h"
00010 #include "item_list.h"
00011 #include "item_queue.h"
00012 #include "item_stack.h"
00013 #include "gadget_list.h"
00014
00018 typedef struct
00019 {
00023     GameMap gameMap;
00027     ItemList todoList;
00031     ItemList inProgressList;
00035     ItemStack bag;
00039     ItemQueue order;
```

```
00043     GadgetList inventory;
00047     int cash;
00051     Location currentLocation;
00052 } State;
00053
00071 State newState(GameMap m, ItemList todo, ItemList inProgress, ItemStack bag, ItemQueue order);
00072
00082 void moveItemToProgressList(State *state, int indexTodo);
00083
00090 void reevaluate(State *state);
00091
00092 #endif
```

Index