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.

4.1.2 Field Documentation

4.1.2.1 colEff

`int colEff`

Banyak kolom matriks.

4.1.2.2 contents

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

2D array untuk menyimpan elemen matriks.

4.1.2.3 rowEff

```
int rowEff
```

Banyak baris matriks.

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

- 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).

4.2.2 Field Documentation

4.2.2.1 id

```
int id
```

Identifier gadget yang unik untuk setiap gadget.

4.2.2.2 name

```
char* name
```

Nama gadget.

4.2.2.3 price

```
int price
```

Harga gadget.

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

- 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.

4.3.2 Field Documentation

4.3.2.1 contents

```
Gadget contents[5]
```

Array statik dengan panjang 5 berisi Gadget.

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

- 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.

4.4.2 Field Documentation

4.4.2.1 _adjacency

```
BooleanMatrix _adjacency
```

4.4.2.2 _locationMatrix

```
LocationMatrix _locationMatrix
```

4.4.2.3 _locations

```
LocationList _locations
```

4.4.2.4 hSize

```
int hSize
```

4.4.2.5 vSize

```
int vSize
```

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

- 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.

4.5.2 Field Documentation

4.5.2.1 dropOffLocation

```
Location dropOffLocation
```

Tempat drop off item.

4.5.2.2 orderTime

```
int orderTime
```

Waktu pesanan item.

4.5.2.3 perishTime

```
int perishTime
```

Waktu hangus item.

4.5.2.4 pickUpLocation

```
Location pickUpLocation
```

Tempat pick up item.

4.5.2.5 type

```
ItemType type
```

Tipe item.

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

- 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.

4.6.2 Field Documentation

4.6.2.1 next

```
ItemListNode* next
```

Pointer ke ItemListNode selanjutnya.

4.6.2.2 value

```
Item value
```

Nilai Item pada ItemListNode ini.

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

- 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.

4.7.2 Field Documentation

4.7.2.1 buffer

```
Item buffer[30]
```

Array tempat menyimpan elemen antrian.

4.7.2.2 headIndex

```
int headIndex
```

Indeks terdepan antrian.

4.7.2.3 tailIndex

```
int tailIndex
```

Indeks terakhir antrian.

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

- 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.

4.8.2 Field Documentation

4.8.2.1 buffer

```
Item buffer[100]
```

Array tempat menyimpan elemen stack.

4.8.2.2 capacity

```
int capacity
```

Kapasitas stack.

4.8.2.3 topIndex

```
int topIndex
```

Indeks teratas stack.

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

- 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.

4.9.2 Field Documentation

4.9.2.1 coordinate

```
Point coordinate
```

Koordinat lokasi.

4.9.2.2 id

```
int id
```

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

4.9.2.3 isDropOffPlace

```
boolean isDropOffPlace
```

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

4.9.2.4 isPickUpPlace

```
boolean isPickUpPlace
```

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

4.9.2.5 isPlayerPlace

```
boolean isPlayerPlace
```

Flag yang menandakan apakah lokasi ini sedang ditempati player.

4.9.2.6 isReachable

```
boolean isReachable
```

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

4.9.2.7 symbol

```
char symbol
```

Simbol lokasi yang dapat ditampilkan.

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

- 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.

4.10.2 Field Documentation

4.10.2.1 buffer

```
Location* buffer
```

Memory tempat menyimpan elemen list.

4.10.2.2 capacity

```
int capacity
```

Kapasitas list.

4.10.2.3 nEff

```
int nEff
```

Banyak elemen list.

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

- 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.

4.11.2 Field Documentation

4.11.2.1 colEff

```
int colEff
```

Banyak kolom matriks yang terdefinisi.

4.11.2.2 contents

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

2D array untuk menyimpan elemen matriks.

4.11.2.3 rowEff

```
int rowEff
```

Banyak baris matriks yang terdefinisi.

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

- 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.

4.12.2 Field Documentation

4.12.2.1 x

```
int x
```

Absis suatu titik.

4.12.2.2 y

```
int y
```

Ordinat suatu titik.

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

- 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.

4.13.2 Field Documentation

4.13.2.1 bag

```
ItemStack bag
```

Tas player.

4.13.2.2 cash

```
int cash
```

Uang player.

4.13.2.3 currentLocation

```
Location currentLocation
```

Lokasi player saat ini.

4.13.2.4 gameMap

```
GameMap gameMap
```

Map dari game yang berjalan.

4.13.2.5 inProgressList

```
ItemList inProgressList
```

In Progress List dari game yang berjalan.

4.13.2.6 inventory

```
GadgetList inventory
```

Inventory player.

4.13.2.7 order

```
ItemQueue order
```

Daftar pesanan dari game yang berjalan.

4.13.2.8 todoList

```
ItemList todoList
```

ToDo List dari game yang berjalan.

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

- state.h

Chapter 5

File Documentation

5.1 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.

5.1.2 Macro Definition Documentation

5.1.2.1 boolean

```
#define boolean unsigned char
```

Tipe boolean.

5.1.2.2 false

```
#define false 0
```

Representasi nilai `false` pada tipe boolean.

5.1.2.3 true

```
#define true 1
```

Representasi nilai `true` pada tipe boolean.

5.2 boolean.h

Go to the documentation of this file.

```
1
6 #ifndef BOOLEAN_H
7 #define BOOLEAN_H
8
13 #define boolean unsigned char
18 #define true 1
23 #define false 0
24
25 #endif
```

5.3 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

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.

5.4 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.

5.4.1 Detailed Description

Header file untuk tipe data BooleanMatrix.

5.4.2 Macro Definition Documentation

5.4.2.1 cols

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

Mengembalikan banyak kolom efektif BooleanMatrix `b`.

Parameters

b	BooleanMatrix instance.

5.4.2.2 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.

5.4.2.3 rows

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

Mengembalikan banyak baris efektif BooleanMatrix `b`.

Parameters

b	BooleanMatrix instance.

5.4.3 Function Documentation

5.4.3.1 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.

5.5 boolean_matrix.h

Go to the documentation of this file.

```
1
6 #ifndef BOOLEAN_MATRIX_H
7 #define BOOLEAN_MATRIX_H
8
9 #include "boolean.h"
10
15 typedef struct
16 {
20     boolean contents[20][30];
24     int rowEff;
28     int colEff;
29 } BooleanMatrix;
30
35 #define rows(b) (b).rowEff
```

```
40 #define cols(b) (b).colEff
47 #define elem(b, i, j) (b).contents[i][j]
48
56 BooleanMatrix newBooleanMatrix(int rows, int cols);
57
58 #endif
```

5.6 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.

5.6.1 Detailed Description

Implementasi tipe data Gadget.

5.6.2 Function Documentation

5.6.2.1 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.

5.6.3 Variable Documentation

5.6.3.1 KAIN_PEMBUNGKUS_WAKTU

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

Instance Gadget "Kain Pembungkus Waktu".

5.6.3.2 MESIN_WAKTU

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

Instance Gadget "Mesin Waktu".

5.6.3.3 NULL_GADGET

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

Instance Gadget yang tidak terdefinisi.

5.6.3.4 PINTU_KEMANA_SAJA

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

Instance Gadget "Pintu Kemana Saja".

5.6.3.5 SENTER_PEMBESAR

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

Instance Gadget "Senter Pembesar".

5.6.3.6 SENTER_PENGECIL

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

Instance Gadget "Senter Pengecil".

5.7 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.

5.7.1 Detailed Description

Header file untuk tipe data Gadget.

5.7.2 Macro Definition Documentation

5.7.2.1 id

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

Mengambil property id dari sebuah Gadget.

Parameters

g	Gadget instance.

5.7.2.2 name

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

Mengambil property name dari sebuah Gadget.

Parameters

g	Gadget instance.

5.7.2.3 price

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

Mengambil property price dari sebuah Gadget.

Parameters

g	Gadget instance.

5.7.3 Function Documentation

5.7.3.1 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.

5.7.4 Variable Documentation

5.7.4.1 KAIN_PEMBUNGKUS_WAKTU

```
Gadget KAIN_PEMBUNGKUS_WAKTU  [extern]
```

Instance Gadget "Kain Pembungkus Waktu".

5.7.4.2 MESIN_WAKTU

```
Gadget MESIN_WAKTU  [extern]
```

Instance Gadget "Mesin Waktu".

5.7.4.3 NULL_GADGET

```
Gadget NULL_GADGET  [extern]
```

Instance Gadget yang tidak terdefinisi.

5.7.4.4 PINTU_KEMANA_SAJA

```
Gadget PINTU_KEMANA_SAJA  [extern]
```

Instance Gadget "Pintu Kemana Saja".

5.7.4.5 SENTER_PEMBESAR

```
Gadget SENTER_PEMBESAR  [extern]
```

Instance Gadget "Senter Pembesar".

5.7.4.6 SENTER_PENGECIL

```
Gadget SENTER_PENGECIL  [extern]
```

Instance Gadget "Senter Pengecil".

5.8 gadget.h

Go to the documentation of this file.

```
1
6 #ifndef GADGET_H
7 #define GADGET_H
8
9 #include "boolean.h"
10
17 typedef struct
18 {
22     int id;
26     int price;
30     char *name;
31 } Gadget;
32
37 #define id(g) (g).id
42 #define price(g) (g).price
47 #define name(g) (g).name
48
52 extern Gadget KAIN_PEMBUNGKUS_WAKTU;
56 extern Gadget SENTER_PEMBESAR;
60 extern Gadget PINTU_KEMANA_SAJA;
64 extern Gadget MESIN_WAKTU;
68 extern Gadget SENTER_PENGECIL;
72 extern Gadget NULL_GADGET;
73
82 boolean isGadgetIdentical(Gadget gadget1, Gadget gadget2);
83
84 #endif
```

5.9 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.

5.9.1 Detailed Description

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

5.9.2 Function Documentation

5.9.2.1 displayGadget()

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

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

Parameters

gList	GadgetList instance.

5.9.2.2 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.

5.9.2.3 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.

5.9.2.4 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.

5.9.2.5 newGadgetList()

```
GadgetList newGadgetList ( )
```

Constructor untuk membuat GadgetList baru.

Returns

Instance GadgetList berisi 5 Gadget yang tidak terdefinisi.

5.9.2.6 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.

5.10 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.

5.10.1 Detailed Description

Header file untuk tipe data GadgetList.

5.10.2 Function Documentation

5.10.2.1 displayGadget()

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

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

Parameters

gList	GadgetList instance.

5.10.2.2 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.

5.10.2.3 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.

5.10.2.4 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.

5.10.2.5 newGadgetList()

```
GadgetList newGadgetList ( )
```

Constructor untuk membuat GadgetList baru.

Returns

Instance GadgetList berisi 5 Gadget yang tidak terdefinisi.

5.10.2.6 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.

5.11 gadget_list.h

Go to the documentation of this file.

```
1
6 #ifndef GADGET_LIST_H
7 #define GADGET_LIST_H
8
9 #include "boolean.h"
10 #include "gadget.h"
11
16 typedef struct
17 {
21     Gadget contents[5];
22 } GadgetList;
23
29 GadgetList newGadgetList();
30
38 boolean isGadgetListEmpty(GadgetList gList);
39
47 boolean isGadgetListFull(GadgetList gList);
48
56 Gadget getGadget(GadgetList gList, int index);
57
65 void setGadget(GadgetList *gList, int index, Gadget g);
66
73 void displayGadget(GadgetList gList);
74
75 #endif
```

5.12 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.

5.12.1 Detailed Description

Implementasi tipe data GameMap.

5.12.2 Function Documentation

5.12.2.1 _getAdjacentLocations()

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

5.12.2.2 displayAdjacentLocation()

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

Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

Parameters

m	GameMap instance.
currentLocation	Lokasi saat ini.

5.12.2.3 displayGameMap()

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

Menampilkan map game ke console output.

Parameters

m	GameMap instance yang akan ditampilkan.

5.12.2.4 getLocationByCoord()

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

Mengambil Location instance berdasarkan koordinat.

Parameters

m	GameMap instance.
p	Koordinat Location.

Returns

Location dengan koordinat p.

5.12.2.5 getLocationById()

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

Mengambil Location instance berdasarkan id.

Parameters

m	GameMap instance.
id	Id Location.

Returns

Location dengan id 'id'.

5.12.2.6 getLocationBySymbol()

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

Mengambil Location instance berdasarkan simbol.

Parameters

m	GameMap instance
symbol	Simbol Location.

Returns

Location dengan simbol 'symbol'.

5.12.2.7 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.

5.12.2.8 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.

5.13 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.

5.13.1 Detailed Description

Header file untuk tipe data GameMap.

5.13.2 Macro Definition Documentation

5.13.2.1 adjMatrix

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

Mengembalikan matriks adjacency dari map m.

See also

BooleanMatrix

Parameters

m	GameMap instance

5.13.2.2 locList

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

Mengembalikan list lokasi yang ada pada m.

See also

LocationList

Parameters

m	GameMap instance

5.13.2.3 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

5.13.2.4 mapLength

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

Mengembalikan panjang GameMap m.

Parameters

m	GameMap instance.

5.13.2.5 mapWidth

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

Mengembalikan lebar GameMap m.

Parameters

m	GameMap instance.

5.13.3 Function Documentation

5.13.3.1 displayAdjacentLocation()

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

Menampilkan lokasi-lokasi yang adjacent terurut berdasarkan koordinat.

Parameters

m	GameMap instance.
currentLocation	Lokasi saat ini.

5.13.3.2 displayGameMap()

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

Menampilkan map game ke console output.

Parameters

m	GameMap instance yang akan ditampilkan.

5.13.3.3 getLocationByCoord()

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

Mengambil Location instance berdasarkan koordinat.

Parameters

m	GameMap instance.
p	Koordinat Location.

Returns

Location dengan koordinat p.

5.13.3.4 getLocationById()

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

Mengambil Location instance berdasarkan id.

Parameters

m	GameMap instance.
id	Id Location.

Returns

Location dengan id 'id'.

5.13.3.5 getLocationBySymbol()

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

Mengambil Location instance berdasarkan simbol.

Parameters

m	GameMap instance
symbol	Simbol Location.

Returns

Location dengan simbol 'symbol'.

5.13.3.6 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.

5.13.3.7 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.

5.14 game_map.h

Go to the documentation of this file.

```
1
6 #ifndef GAMEMAP_H
7 #define GAMEMAP_H
8
9 #include "location.h"
10 #include "boolean_matrix.h"
11 #include "location_list.h"
12 #include "location_matrix.h"
13
19 typedef struct
20 {
21     int hSize;
22     int vSize;
23     BooleanMatrix _adjacency;
24     LocationList _locations;
25     LocationMatrix _locationMatrix;
26 } GameMap;
27
37 GameMap newGameMap(int hSize, int vSize, BooleanMatrix adjMatrix, Location *locations);
38
43 #define mapLength(m) (m).hSize
48 #define mapWidth(m) (m).vSize
55 #define adjMatrix(m) (m)._adjacency
62 #define locList(m) (m)._locations
70 #define locMatrix(m) (m)._locationMatrix
71
77 void displayGameMap(GameMap m);
78
86 void displayAdjacentLocation(GameMap m, Location currentLocation);
87
95 boolean isAdjacentTo(GameMap m, Location a, Location b);
96
104 Location getLocationById(GameMap m, int id);
105
113 Location getLocationBySymbol(GameMap m, char symbol);
114
122 Location getLocationByCoord(GameMap m, Point p);
123
124 #endif
```

5.15 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.

5.15.1 Detailed Description

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

5.15.2 Function Documentation

5.15.2.1 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.

5.15.2.2 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.

5.15.2.3 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.

5.15.2.4 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.

5.15.2.5 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.

5.15.2.6 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.

5.16 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.

5.16.1 Detailed Description

Header file untuk tipe data Item.

5.16.2 Macro Definition Documentation

5.16.2.1 dropOffLoc

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

Mengambil lokasi drop off item.

Parameters

item	Item instance.

5.16.2.2 HEAVY

```
#define HEAVY 1
```

Tipe item Heavy.

5.16.2.3 itemType

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

Mengambil tipe item.

Parameters

item	Item instance.

5.16.2.4 NORMAL

```
#define NORMAL 0
```

Tipe item Normal.

5.16.2.5 orderTime

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

Mengambil waktu order item.

Parameters

item	Item instance.

5.16.2.6 PERISHABLE

```
#define PERISHABLE 2
```

Tipe item Perishable.

5.16.2.7 perishTime

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

Mengambil waktu hangus item.

Parameters

item	Item instance.

5.16.2.8 pickUpLoc

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

Mengambil lokasi pick up item.

Parameters

item	Item instance.

5.16.2.9 UNTIMED

```
#define UNTIMED -1
```

Perish time untuk item yang non-perishable.

5.16.2.10 VIP

```
#define VIP 3
```

Tipe item VIP.

5.16.3 Typedef Documentation

5.16.3.1 ItemType

```
typedef int ItemType
```

Alias untuk tipe ItemType.

5.16.4 Function Documentation

5.16.4.1 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.

5.16.4.2 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.

5.16.4.3 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.

5.16.4.4 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.

5.16.4.5 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.

5.16.4.6 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.

5.17 item.h

Go to the documentation of this file.

```
1
6 #ifndef ITEM_H
7 #define ITEM_H
8
9 #include "location.h"
10
14 typedef int ItemType;
18 #define NORMAL 0
22 #define HEAVY 1
26 #define PERISHABLE 2
30 #define VIP 3
31
36 #define UNTIMED -1
37
42 typedef struct
43 {
47     int orderTime;
51     Location pickUpLocation;
55     Location dropOffLocation;
59     ItemType type;
63     int perishTime;
64 } Item;
65
70 #define pickUpLoc(item) (item).pickUpLocation
75 #define dropOffLoc(item) (item).dropOffLocation
80 #define itemType(item) (item).type
85 #define perishTime(item) (item).perishTime
90 #define orderTime(item) (item).orderTime
91
102 Item newItem(int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int
       perishTime);
103
111 boolean isItemIdentical(Item item1, Item item2);
112
120 boolean isNormalItem(Item item);
121
129 boolean isHeavyItem(Item item);
130
138 boolean isPerishableItem(Item item);
139
147 boolean isVIPItem(Item item);
148
149 #endif
```

5.18 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.

5.18.1 Detailed Description

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

5.18.2 Function Documentation

5.18.2.1 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.

5.18.2.2 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.

5.18.2.3 deleteItemLast()

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

Mengambil & menghapus Item terakhir iList.

Parameters

iList	ItemList instance.
item	Item di posisi terakhir iList.

5.18.2.4 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.

5.18.2.5 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.

5.18.2.6 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.

5.18.2.7 insertItemFirst()

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

Insert Item di awal list iList.

Parameters

iList	ItemList instance.
item	Item instance.

5.18.2.8 insertItemLast()

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

Insert Item di akhir list iList.

Parameters

iList	ItemList instance.
item	Item instance.

5.18.2.9 isItemListEmpty()

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

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

iList	ItemList instance.

Returns

true jika iList kosong, false selainnya.

5.18.2.10 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.

5.18.2.11 itemListLength()

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

Mengembalikan panjang suatu ItemList.

Parameters

iList	ItemList instance.

Returns

Panjang dari iList.

5.18.2.12 newItemList()

```
ItemList newItemList ( )
```

Constructor untuk membuat ItemList baru.

Returns

ItemList kosong.

5.18.2.13 newItemListNode()

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

Constructor untuk membuat ItemListNode baru.

Parameters

item	Value yang di-hold oleh node ini.

Returns

ItemListNode instance berisi item.

5.18.2.14 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.

5.19 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.

5.19.1 Detailed Description

Header file untuk tipe data ItemList.

5.19.2 Macro Definition Documentation

5.19.2.1 next

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

Mengambil pointer ke next node dari sebuah ItemListNode.

Parameters

node	ItemListNode instance.

5.19.2.2 value

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

Mengambil value dari sebuah ItemListNode.

Parameters

node	ItemListNode instance.

5.19.3 Typedef Documentation

5.19.3.1 ItemList

```
typedef ItemListNode* ItemList
```

Alias untuk tipe data ItemListNode.

5.19.4 Function Documentation

5.19.4.1 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.

5.19.4.2 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.

5.19.4.3 deleteItemLast()

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

Mengambil & menghapus Item terakhir iList.

Parameters

iList	ItemList instance.
item	Item di posisi terakhir iList.

5.19.4.4 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.

5.19.4.5 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.

5.19.4.6 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.

5.19.4.7 insertItemFirst()

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

Insert Item di awal list iList.

Parameters

iList	ItemList instance.
item	Item instance.

5.19.4.8 insertItemLast()

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

Insert Item di akhir list iList.

Parameters

iList	ItemList instance.
item	Item instance.

5.19.4.9 isItemListEmpty()

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

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

iList	ItemList instance.

Returns

true jika iList kosong, false selainnya.

5.19.4.10 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.

5.19.4.11 itemListLength()

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

Mengembalikan panjang suatu ItemList.

Parameters

iList	ItemList instance.

Returns

Panjang dari iList.

5.19.4.12 newItemList()

```
ItemList newItemList ( )
```

Constructor untuk membuat ItemList baru.

Returns

ItemList kosong.

5.19.4.13 newItemListNode()

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

Constructor untuk membuat ItemListNode baru.

Parameters

item	Value yang di-hold oleh node ini.

Returns

ItemListNode instance berisi item.

5.19.4.14 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.

5.20 item_list.h

Go to the documentation of this file.

```
1
6 #ifndef ITEM_DYNAMIC_LIST_H
7 #define ITEM_DYNAMIC_LIST_H
8
9 #include "boolean.h"
10 #include "item.h"
11
15 typedef ItemListNode *ItemList;
16
21 typedef struct
22 {
26     Item value;
30     ItemListNode *next;
31 } ItemListNode;
32
37 #define value(node) (node).value
42 #define next(node) (node).next
43
49 ItemList newItemList();
50
57 ItemListNode newItemListNode(Item item);
58
65 boolean isItemListEmpty(ItemList iList);
66
75 boolean isItemListIndexValid(ItemList iList, int index);
76
83 int itemListLength(ItemList iList);
84
93 int indexOfItem(ItemList iList, Item item);
94
102 Item getItem(ItemList iList, int index);
103
111 void setItem(ItemList *iList, int index, Item item);
112
119 void insertItemFirst(ItemList *iList, Item item);
120
128 void insertItemAt(ItemList *iList, int index, Item item);
129
136 void insertItemLast(ItemList *iList, Item item);
137
144 void deleteItemFirst(ItemList *iList, Item *item);
145
153 void deleteItemAt(ItemList *iList, int index, Item *item);
154
161 void deleteItemLast(ItemList *iList, Item *item);
162
163 #endif
```

5.21 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.

5.21.1 Detailed Description

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

5.21.2 Function Documentation

5.21.2.1 dequeue()

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

Mengambil Item terdepan pada antrian q.

Parameters

q	ItemQueue instance.
item	Item terdepan pada antrian q.

5.21.2.2 enqueue()

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

Menambah Item item pada antrian q.

Parameters

q	ItemQueue instance.
item	Item instance.

5.21.2.3 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.

5.21.2.4 newItemQueue()

```
ItemQueue newItemQueue ( )
```

Constructor untuk membuat ItemQueue baru.

Returns

ItemQueue instance baru yang kosong.

5.21.2.5 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.

5.22 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.

5.22.1 Detailed Description

Header file untuk tipe data ItemQueue.

5.22.2 Macro Definition Documentation

5.22.2.1 head

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

Mengambil head Item pada antrian q.

Parameters

q	ItemQueue instance.

5.22.2.2 headIndex

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

Mengambil indeks head pada antrian q.

Parameters

q	ItemQueue instance.

5.22.2.3 tail

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

Mengambil tail Item pada antrian q.

Parameters

q	ItemQueue instance.

5.22.2.4 tailIndex

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

Mengambil indeks tail pada antrian q.

Parameters

q	ItemQueue instance.

5.22.3 Function Documentation

5.22.3.1 dequeue()

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

Mengambil Item terdepan pada antrian q.

Parameters

q	ItemQueue instance.
item	Item terdepan pada antrian q.

5.22.3.2 enqueue()

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

Menambah Item item pada antrian q.

Parameters

q	ItemQueue instance.
item	Item instance.

5.22.3.3 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.

5.22.3.4 newItemQueue()

```
ItemQueue newItemQueue ( )
```

Constructor untuk membuat ItemQueue baru.

Returns

ItemQueue instance baru yang kosong.

5.22.3.5 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.

5.23 item_queue.h

Go to the documentation of this file.

```
1
6 #ifndef ITEM_QUEUE_H
7 #define ITEM_QUEUE_H
8
9 #include "boolean.h"
10 #include "item.h"
11
17 typedef struct
18 {
22     int headIndex;
26     int tailIndex;
31     Item buffer[30];
32 } ItemQueue;
33
38 #define headIndex(q) (q).headIndex
43 #define tailIndex(q) (q).tailIndex
48 #define head(q) (q).buffer[(q).headIndex]
53 #define tail(q) (q).buffer[(q).tailIndex]
54
60 ItemQueue newItemQueue();
61
70 int peekHeadTime(ItemQueue q);
71
78 boolean isEmpty(ItemQueue q);
```

```
79
86 void enqueue(ItemQueue *q, Item item);
87
94 void dequeue(ItemQueue *q, Item *item);
95
96 #endif
```

5.24 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.

5.24.1 Detailed Description

Implementasi tipe data ItemStack. Hanya digunakan untuk INVENTORY.

5.24.2 Function Documentation

5.24.2.1 _clampCapacity()

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

Membatasi kapasitas stack jika kapasitas melebihi batas.

Parameters

stack	ItemStack instance.

5.24.2.2 doubleCapacity()

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

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

Parameters

stack	ItemStack instance.

5.24.2.3 incrementCapacity()

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

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

Parameters

stack	ItemStack instance.

5.24.2.4 isStackEmpty()

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

Mengecek apakah stack kosong atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack kosong, false selainnya.

5.24.2.5 isStackFull()

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

Mengecek apakah stack penuh atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack penuh, false selainnya.

5.24.2.6 newItemStack()

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

Constructor untuk membuat ItemStack baru.

Parameters

capacity	Kapasitas stack.

Returns

ItemStack instace baru yang kosong.

5.24.2.7 pop()

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

Mengambil item dari atas stack.

Parameters

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

5.24.2.8 push()

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

Memasukkan item ke atas stack.

Parameters

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

5.25 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.

5.25.1 Detailed Description

Header file untuk tipe data ItemStack.

5.25.2 Macro Definition Documentation

5.25.2.1 capacity

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

Mengambil kapasitas stack s.

Parameters

s	ItemStack instance.

5.25.2.2 ITEM_STACK_MAX_CAPACITY

```
#define ITEM_STACK_MAX_CAPACITY 100
```

Kapasitas maksimum ItemStack.

5.25.2.3 top

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

Mengambil Item teratas pada stack s.

Parameters

s	ItemStack instance.

5.25.2.4 topIndex

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

Mengambil indeks teratas stack s.

Parameters

s	ItemStack instance.

5.25.3 Function Documentation

5.25.3.1 _clampCapacity()

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

Membatasi kapasitas stack jika kapasitas melebihi batas.

Parameters

stack	ItemStack instance.

5.25.3.2 doubleCapacity()

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

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

Parameters

stack	ItemStack instance.

5.25.3.3 incrementCapacity()

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

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

Parameters

stack	ItemStack instance.

5.25.3.4 isStackEmpty()

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

Mengecek apakah stack kosong atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack kosong, false selainnya.

5.25.3.5 isStackFull()

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

Mengecek apakah stack penuh atau tidak.

Parameters

stack	ItemStack instance.

Returns

true jika stack penuh, false selainnya.

5.25.3.6 newItemStack()

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

Constructor untuk membuat ItemStack baru.

Parameters

capacity	Kapasitas stack.

Returns

ItemStack instace baru yang kosong.

5.25.3.7 pop()

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

Mengambil item dari atas stack.

Parameters

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

5.25.3.8 push()

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

Memasukkan item ke atas stack.

Parameters

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

5.26 item_stack.h

Go to the documentation of this file.

```
1
6 #ifndef ITEM_STACK_H
7 #define ITEM_STACK_H
8
9 #include "boolean.h"
10 #include "item.h"
11
16 typedef struct
17 {
21     int topIndex;
25     int capacity;
29     Item buffer[100];
30 } ItemStack;
31
35 #define ITEM_STACK_MAX_CAPACITY 100
40 #define topIndex(s) (s).topIndex
45 #define top(s) (s).buffer[(s).topIndex]
50 #define capacity(s) (s).capacity
51
58 ItemStack newItemStack(int capacity);
59
66 boolean isStackEmpty(ItemStack stack);
67
74 boolean isStackFull(ItemStack stack);
75
82 void push(ItemStack *stack, Item item);
83
90 void pop(ItemStack *stack, Item *item);
91
99 void incrementCapacity(ItemStack *stack);
100
108 void doubleCapacity(ItemStack *stack);
109
116 void _clampCapacity(ItemStack *stack);
117
118 #endif
```

5.27 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.

5.27.1 Detailed Description

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

See also

GameMap

5.27.2 Function Documentation

5.27.2.1 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.

5.27.2.2 isLocationDefined()

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

Mengecek apakah suatu lokasi terdefinisi atau tidak.

Parameters

l	Location instance.

Returns

true jika l terdefinisi, false selainnya.

5.27.2.3 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.

5.27.2.4 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.

5.27.2.5 setAsDropOffPlace()

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

Set lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

5.27.2.6 setAsPickUpPlace()

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

Set lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

5.27.2.7 setAsPlayerPlace()

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

Set lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.27.2.8 setAsReachable()

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

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

Parameters

l	Location instance.

5.27.2.9 toggleAsPlayerPlace()

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

Toggle lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.27.2.10 unsetAsDropOffPlace()

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

Unset lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

5.27.2.11 unsetAsPickUpPlace()

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

Unset lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

5.27.2.12 unsetAsPlayerPlace()

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

Unset lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.27.2.13 unsetAsReachable()

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

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

Parameters

l	Location instance.

5.27.2.14 writeLocationSymbol()

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

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

Parameters

l	Location instance.

5.27.3 Variable Documentation

5.27.3.1 NULL_LOCATION

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

Location yang tidak terdefinisi.

5.28 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.

5.28.1 Detailed Description

Header file untuk tipe data Location.

5.28.2 Macro Definition Documentation

5.28.2.1 coord

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

Mengambil koordinat lokasi.

Parameters

l	Location instance.

5.28.2.2 id

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

Mengambil id lokasi.

Parameters

l	Location instance.

5.28.2.3 symbol

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

Mengambil simbol lokasi.

Parameters

l	Location instance.

5.28.3 Function Documentation

5.28.3.1 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.

5.28.3.2 isLocationDefined()

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

Mengecek apakah suatu lokasi terdefinisi atau tidak.

Parameters

l	Location instance.

Returns

true jika l terdefinisi, false selainnya.

5.28.3.3 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.

5.28.3.4 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.

5.28.3.5 setAsDropOffPlace()

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

Set lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

5.28.3.6 setAsPickUpPlace()

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

Set lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

5.28.3.7 setAsPlayerPlace()

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

Set lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.28.3.8 setAsReachable()

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

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

Parameters

l	Location instance.

5.28.3.9 toggleAsPlayerPlace()

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

Toggle lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.28.3.10 unsetAsDropOffPlace()

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

Unset lokasi sebagai tempat drop off Item.

Parameters

l	Location instance.

5.28.3.11 unsetAsPickUpPlace()

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

Unset lokasi sebagai tempat pick up Item.

Parameters

l	Location instance.

5.28.3.12 unsetAsPlayerPlace()

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

Unset lokasi sebagai lokasi player.

Parameters

l	Location instance.

5.28.3.13 unsetAsReachable()

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

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

Parameters

l	Location instance.

5.28.3.14 writeLocationSymbol()

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

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

Parameters

l	Location instance.

5.28.4 Variable Documentation

5.28.4.1 NULL_LOCATION

```
Location NULL_LOCATION  [extern]
```

Location yang tidak terdefinisi.

5.29 location.h

Go to the documentation of this file.

```
1
6 #ifndef LOCATION_H
7 #define LOCATION_H
8
9 #include "boolean.h"
10 #include "point.h"
11
15 extern Location NULL_LOCATION;
16
22 typedef struct
23 {
28     int id;
32     char symbol;
36     Point coordinate;
41     boolean isPlayerPlace;
46     boolean isPickUpPlace;
51     boolean isDropOffPlace;
56     boolean isReachable;
57 } Location;
58
63 #define id(l) (l).id
68 #define symbol(l) (l).symbol
73 #define coord(l) (l).coordinate
74
83 Location newLocation(int id, char symbol, Point coordinate);
84
93 boolean isAt(Location l, Point p);
94
103 boolean isLocationIdentical(Location l1, Location l2);
104
112 boolean isLocationDefined(Location l);
113
120 void writeLocationSymbol(Location l);
121
127 void setAsPickUpPlace(Location *l);
128
134 void unsetAsPickUpPlace(Location *l);
135
141 void setAsDropOffPlace(Location *l);
142
148 void unsetAsDropOffPlace(Location *l);
149
156 void setAsReachable(Location *l);
157
164 void unsetAsReachable(Location *l);
165
171 void setAsPlayerPlace(Location *l);
172
178 void unsetAsPlayerPlace(Location *l);
179
185 void toggleAsPlayerPlace(Location *l);
186
187 #endif
```

5.30 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.

5.30.1 Detailed Description

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

5.30.2 Function Documentation

5.30.2.1 _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.

5.30.2.2 _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.

5.30.2.3 _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.

5.30.2.4 compactList()

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

Merapatkan list l.

Parameters

l	LocationList instance.

5.30.2.5 dealocateLocationList()

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

Menghapus memory list dari heap memory.

Parameters

l	LocationList instance.

5.30.2.6 deleteLast()

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

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	l	LocaitonList instance.
out	*location*	Location instance.

5.30.2.7 growList()

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

Menambah kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan ditambah.

5.30.2.8 insertLast()

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

Memasukkan Location ke akhir list.

Parameters

l	LocationList instance.
location	Location instance.

5.30.2.9 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.

5.30.2.10 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.

5.30.2.11 isLocationListEmpty()

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

Mengecek apakah list l kosong atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l kosong, false selainnya.

5.30.2.12 isLocationListFull()

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

Mengecek apakah list l penuh atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l penuh, false selainnya.

5.30.2.13 length()

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

Mengembalikan panjang list l.

Parameters

l	LocationList instance.

Returns

Panjang list l.

5.30.2.14 newLocationList()

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

Constructor untuk membuat LocationList baru.

Parameters

capacity	Kapasitas list.

Returns

LocationList instance baru yang kosong.

5.30.2.15 shrinkList()

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

Mengurangi kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan dikurangi.

5.30.2.16 sortLocationListByCoord()

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

Mengurutkan Location dalam l berdasarkan koordinat.

See also

Location

Parameters

l	LocationList instance.

5.31 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.

5.31.1 Detailed Description

Header file untuk tipe data LocationList.

5.31.2 Macro Definition Documentation

5.31.2.1 buffer

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

Mengambil memory list.

Parameters

l	LocationList instance.

5.31.2.2 capacity

```
#define capacity(
```

```
            l ) (l).capacity
```

Mengambil kapasitas elemen list.

Parameters

l	LocationList instance.

5.31.2.3 lElem

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

Mengambil elemen list pada indeks tertentu.

Parameters

l	LocationList instance.
i	Indeks elemen yang akan diambil.

5.31.2.4 neff

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

Mengambil banyak elemen list.

Parameters

l	LocationList instance.

5.31.3 Function Documentation

5.31.3.1 _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.

5.31.3.2 _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.

5.31.3.3 _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.

5.31.3.4 compactList()

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

Merapatkan list l.

Parameters

l	LocationList instance.

5.31.3.5 dealocateLocationList()

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

Menghapus memory list dari heap memory.

Parameters

l	LocationList instance.

5.31.3.6 deleteLast()

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

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	l	LocaitonList instance.
out	*location*	Location instance.

5.31.3.7 growList()

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

Menambah kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan ditambah.

5.31.3.8 insertLast()

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

Memasukkan Location ke akhir list.

Parameters

l	LocationList instance.
location	Location instance.

5.31.3.9 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.

5.31.3.10 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.

5.31.3.11 isLocationListEmpty()

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

Mengecek apakah list l kosong atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l kosong, false selainnya.

5.31.3.12 isLocationListFull()

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

Mengecek apakah list l penuh atau tidak.

Parameters

l	LocationList instance.

Returns

true jika list l penuh, false selainnya.

5.31.3.13 length()

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

Mengembalikan panjang list l.

Parameters

l	LocationList instance.

Returns

Panjang list l.

5.31.3.14 newLocationList()

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

Constructor untuk membuat LocationList baru.

Parameters

capacity	Kapasitas list.

Returns

LocationList instance baru yang kosong.

5.31.3.15 shrinkList()

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

Mengurangi kapasitas list l sebanyak num.

Parameters

l	LocationList instance.
num	Banyak kapasitas yang akan dikurangi.

5.31.3.16 sortLocationListByCoord()

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

Mengurutkan Location dalam l berdasarkan koordinat.

See also

Location

Parameters

l	LocationList instance.

5.32 location_list.h

Go to the documentation of this file.

```
1
6 #ifndef LOCATION_LIST_H
7 #define LOCATION_LIST_H
8
9 #include "boolean.h"
10 #include "location.h"
11 #include "point.h"
12
17 typedef struct
18 {
22     Location *buffer;
26     int nEff;
30     int capacity;
31 } LocationList;
32
37 #define neff(l) (l).nEff
42 #define capacity(l) (l).capacity
47 #define buffer(l) (l).buffer
53 #define lElem(l, i) (l).buffer[i]
54
61 LocationList newLocationList(int capacity);
62
68 void dealocateLocationList(LocationList *l);
69
76 int length(LocationList l);
77
86 boolean isIndexValid(LocationList l, int i);
87
96 boolean isIndexEff(LocationList l, int i);
97
104 boolean isLocationListEmpty(LocationList l);
105
112 boolean isLocationListFull(LocationList l);
113
120 void insertLast(LocationList *l, Location location);
121
129 void deleteLast(LocationList *l, Location *location);
130
137 void growList(LocationList *l, int num);
138
145 void shrinkList(LocationList *l, int num);
146
152 void compactList(LocationList *l);
153
160 void sortLocationListByCoord(LocationList *l);
161
171 Location _getLocationById(LocationList l, int id);
172
182 Location _getLocationBySymbol(LocationList l, char symbol);
183
193 Location _getLocationByCoord(LocationList l, Point p);
194
195 #endif
```

5.33 location_matrix.c File Reference

Implementasi tipe data LocationMatrix.

```
#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.

5.33.1 Detailed Description

Implementasi tipe data LocationMatrix.

5.33.2 Function Documentation

5.33.2.1 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.

5.33.2.2 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.

5.34 location_matrix.h File Reference

Header file untuk 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.

5.34.1 Detailed Description

Header file untuk tipe data LocationMatrix.

5.34.2 Macro Definition Documentation

5.34.2.1 cols

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

Mengambil banyak kolom matriks l.

Parameters

l	LocationMatrix instance.

5.34.2.2 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.

5.34.2.3 rows

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

Mengambil banyak baris matriks l.

Parameters

l	LocationMatrix instance.

5.34.3 Function Documentation

5.34.3.1 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.

5.34.3.2 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.

5.35 location_matrix.h

Go to the documentation of this file.

```
1
6 #ifndef LOCATION_MATRIX_H
7 #define LOCATION_MATRIX_H
8
9 #include "boolean.h"
10 #include "location.h"
11
16 typedef struct
17 {
21     Location contents[20][30];
25     int rowEff;
29     int colEff;
30 } LocationMatrix;
31
36 #define rows(l) (l).rowEff
41 #define cols(l) (l).colEff
49 #define elem(l, i, j) (l).contents[i][j]
50
58 LocationMatrix newLocationMatrix(int rows, int cols);
59
69 void lSetElem(LocationMatrix *l, int rowIndex, int colIndex, Location value);
70
71 #endif
```

5.36 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.

5.36.1 Detailed Description

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

See also

Location

5.36.2 Function Documentation

5.36.2.1 displayPoint()

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

Menuliskan titik p ke console output.

Parameters

p	Point instance.

5.36.2.2 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.

5.36.2.3 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.

5.36.2.4 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).

5.37 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.

5.37.1 Detailed Description

Header file untuk tipe data Point.

5.37.2 Macro Definition Documentation

5.37.2.1 abs

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

Mengambil absis dari suatu titik.

Parameters

p	Point instance.

5.37.2.2 ord

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

Mengambil ordinat dari suatu titik.

Parameters

p	Point instance.

5.37.3 Function Documentation

5.37.3.1 displayPoint()

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

Menuliskan titik p ke console output.

Parameters

p	Point instance.

5.37.3.2 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.

5.37.3.3 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.

5.37.3.4 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).

5.38 point.h

Go to the documentation of this file.

```
1
6 #ifndef POINT_H
7 #define POINT_H
8
```

```
9 #include "boolean.h"
10
15 typedef struct
16 {
20     int x;
24     int y;
25 } Point;
26
34 Point newPoint(int x, int y);
35
40 #define abs(p) (p).x
45 #define ord(p) (p).y
46
54 boolean isPointIdentical(Point p1, Point p2);
55
66 boolean isPointBefore(Point p1, Point p2);
67
73 void displayPoint(Point p);
74
75 #endif
```

5.39 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.

5.39.1 Detailed Description

Header file untuk State game.

5.39.2 Function Documentation

5.39.2.1 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.

5.39.2.2 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

5.39.2.3 reevaluate()

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

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

Parameters

state	State saat ini.

5.40 state.h

Go to the documentation of this file.

```
1
6 #ifndef STATE_H
7 #define STATE_H
8
9 #include "game_map.h"
10 #include "item_list.h"
11 #include "item_queue.h"
12 #include "item_stack.h"
13 #include "gadget_list.h"
14
18 typedef struct
19 {
23     GameMap gameMap;
27     ItemList todoList;
31     ItemList inProgressList;
35     ItemStack bag;
39     ItemQueue order;
43     GadgetList inventory;
47     int cash;
51     Location currentLocation;
52 } State;
53
71 State newState(GameMap m, ItemList todo, ItemList inProgress, ItemStack bag, ItemQueue order);
72
82 void moveItemToProgressList(State *state, int indexTodo);
83
90 void reevaluate(State *state);
91
92 #endif
```

Index