

Mobilita

Generated by Doxygen 1.9.2

1 Todo List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 BooleanMatrix Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 colEff	7
4.1.2.2 contents	8
4.1.2.3 rowEff	8
4.2 Gadget Struct Reference	8
4.2.1 Detailed Description	8
4.2.2 Field Documentation	8
4.2.2.1 id	9
4.2.2.2 name	9
4.2.2.3 price	9
4.3 GadgetList Struct Reference	9
4.3.1 Detailed Description	9
4.3.2 Field Documentation	9
4.3.2.1 contents	10
4.4 GameMap Struct Reference	10
4.4.1 Detailed Description	10
4.4.2 Field Documentation	10
4.4.2.1 _adjacency	10
4.4.2.2 _locationMatrix	10
4.4.2.3 _locations	11
4.4.2.4 hSize	11
4.4.2.5 vSize	11
4.5 Item Struct Reference	11
4.5.1 Detailed Description	11
4.5.2 Field Documentation	11
4.5.2.1 dropOffLocation	12
4.5.2.2 orderTime	12
4.5.2.3 perishTime	12
4.5.2.4 pickUpLocation	12
4.5.2.5 type	12
4.6 ItemListNode Struct Reference	12
4.6.1 Detailed Description	13

4.6.2 Field Documentation	13
4.6.2.1 next	13
4.6.2.2 value	13
4.7 ItemQueue Struct Reference	13
4.7.1 Detailed Description	14
4.7.2 Field Documentation	14
4.7.2.1 buffer	14
4.7.2.2 headIndex	14
4.7.2.3 tailIndex	14
4.8 ItemStack Struct Reference	14
4.8.1 Detailed Description	15
4.8.2 Field Documentation	15
4.8.2.1 buffer	15
4.8.2.2 capacity	15
4.8.2.3 topIndex	15
4.9 Location Struct Reference	15
4.9.1 Detailed Description	16
4.9.2 Field Documentation	16
4.9.2.1 coordinate	16
4.9.2.2 id	16
4.9.2.3 isDropOffPlace	16
4.9.2.4 isPickUpPlace	17
4.9.2.5 isPlayerPlace	17
4.9.2.6 isReachable	17
4.9.2.7 symbol	17
4.10 LocationList Struct Reference	17
4.10.1 Detailed Description	18
4.10.2 Field Documentation	18
4.10.2.1 buffer	18
4.10.2.2 capacity	18
4.10.2.3 nEff	18
4.11 LocationMatrix Struct Reference	18
4.11.1 Detailed Description	19
4.11.2 Field Documentation	19
4.11.2.1 colEff	19
4.11.2.2 contents	19
4.11.2.3 rowEff	19
4.12 Point Struct Reference	19
4.12.1 Detailed Description	20
4.12.2 Field Documentation	20
4.12.2.1 x	20
4.12.2.2 y	20

4.13 State Struct Reference	20
4.13.1 Detailed Description	21
4.13.2 Field Documentation	21
4.13.2.1 bag	21
4.13.2.2 cash	21
4.13.2.3 currentLocation	21
4.13.2.4 gameMap	21
4.13.2.5 inProgressList	21
4.13.2.6 inventory	22
4.13.2.7 order	22
4.13.2.8 todoList	22
5 File Documentation	23
5.1 boolean.h File Reference	23
5.1.1 Detailed Description	23
5.1.2 Macro Definition Documentation	23
5.1.2.1 boolean	23
5.1.2.2 false	24
5.1.2.3 true	24
5.2 boolean.h	24
5.3 boolean_matrix.c File Reference	24
5.3.1 Detailed Description	24
5.3.2 Function Documentation	25
5.3.2.1 newBooleanMatrix()	25
5.4 boolean_matrix.h File Reference	25
5.4.1 Detailed Description	26
5.4.2 Macro Definition Documentation	26
5.4.2.1 cols	26
5.4.2.2 elem	26
5.4.2.3 rows	26
5.4.3 Function Documentation	27
5.4.3.1 newBooleanMatrix()	27
5.5 boolean_matrix.h	27
5.6 gadget.c File Reference	28
5.6.1 Detailed Description	28
5.6.2 Function Documentation	28
5.6.2.1 isGadgetIdentical()	28
5.6.3 Variable Documentation	29
5.6.3.1 KAIN_PEMBUNGKUS_WAKTU	29
5.6.3.2 MESIN_WAKTU	29
5.6.3.3 NULL_GADGET	29
5.6.3.4 PINTU_KEMANA_SAJA	29

5.6.3.5 SENTER_PEMBESAR	30
5.6.3.6 SENTER_PENGECIL	30
5.7 gadget.h File Reference	30
5.7.1 Detailed Description	31
5.7.2 Macro Definition Documentation	31
5.7.2.1 id	31
5.7.2.2 name	31
5.7.2.3 price	32
5.7.3 Function Documentation	32
5.7.3.1 isGadgetIdentical()	32
5.7.4 Variable Documentation	32
5.7.4.1 KAIN_PEMBUNGKUS_WAKTU	32
5.7.4.2 MESIN_WAKTU	33
5.7.4.3 NULL_GADGET	33
5.7.4.4 PINTU_KEMANA_SAJA	33
5.7.4.5 SENTER_PEMBESAR	33
5.7.4.6 SENTER_PENGECIL	33
5.8 gadget.h	34
5.9 gadget_list.c File Reference	34
5.9.1 Detailed Description	34
5.9.2 Function Documentation	35
5.9.2.1 displayGadget()	35
5.9.2.2 getGadget()	35
5.9.2.3 isGadgetListEmpty()	35
5.9.2.4 isGadgetListFull()	36
5.9.2.5 newGadgetList()	36
5.9.2.6 setGadget()	36
5.10 gadget_list.h File Reference	37
5.10.1 Detailed Description	37
5.10.2 Function Documentation	37
5.10.2.1 displayGadget()	37
5.10.2.2 getGadget()	38
5.10.2.3 isGadgetListEmpty()	38
5.10.2.4 isGadgetListFull()	39
5.10.2.5 newGadgetList()	39
5.10.2.6 setGadget()	39
5.11 gadget_list.h	40
5.12 game_map.c File Reference	40
5.12.1 Detailed Description	41
5.12.2 Function Documentation	41
5.12.2.1 _getAdjacentLocations()	41
5.12.2.2 displayAdjacentLocation()	41

5.12.2.3 displayGameMap()	41
5.12.2.4 getLocationByCoord()	42
5.12.2.5 getLocationById()	42
5.12.2.6 getLocationBySymbol()	42
5.12.2.7 isAdjacentTo()	43
5.12.2.8 newGameMap()	43
5.13 game_map.h File Reference	43
5.13.1 Detailed Description	44
5.13.2 Macro Definition Documentation	44
5.13.2.1 adjMatrix	45
5.13.2.2 locList	45
5.13.2.3 locMatrix	45
5.13.2.4 mapLength	46
5.13.2.5 mapWidth	46
5.13.3 Function Documentation	46
5.13.3.1 displayAdjacentLocation()	46
5.13.3.2 displayGameMap()	47
5.13.3.3 getLocationByCoord()	47
5.13.3.4 getLocationById()	47
5.13.3.5 getLocationBySymbol()	48
5.13.3.6 isAdjacentTo()	48
5.13.3.7 newGameMap()	48
5.14 game_map.h	49
5.15 item.c File Reference	49
5.15.1 Detailed Description	50
5.15.2 Function Documentation	50
5.15.2.1 isHeavyItem()	50
5.15.2.2 isItemIdentical()	50
5.15.2.3 isNormalItem()	51
5.15.2.4 isPerishableItem()	51
5.15.2.5 isVIPItem()	51
5.15.2.6 newItem()	52
5.16 item.h File Reference	52
5.16.1 Detailed Description	53
5.16.2 Macro Definition Documentation	54
5.16.2.1 dropOffLoc	54
5.16.2.2 HEAVY	54
5.16.2.3 itemType	54
5.16.2.4 NORMAL	54
5.16.2.5 orderTime	55
5.16.2.6 PERISHABLE	55
5.16.2.7 perishTime	55

5.16.2.8 pickUpLoc	55
5.16.2.9 UNTIMED	56
5.16.2.10 VIP	56
5.16.3 Typedef Documentation	56
5.16.3.1 ItemType	56
5.16.4 Function Documentation	56
5.16.4.1 isHeavyItem()	56
5.16.4.2 isItemIdentical()	57
5.16.4.3 isNormalItem()	57
5.16.4.4 isPerishableItem()	57
5.16.4.5 isVIPItem()	58
5.16.4.6 newItem()	58
5.17 item.h	58
5.18 item_list.c File Reference	59
5.18.1 Detailed Description	60
5.18.2 Function Documentation	60
5.18.2.1 deleteItemAt()	60
5.18.2.2 deleteItemFirst()	60
5.18.2.3 deleteItemLast()	61
5.18.2.4 getItem()	61
5.18.2.5 indexOfItem()	61
5.18.2.6 insertItemAt()	62
5.18.2.7 insertItemFirst()	62
5.18.2.8 insertItemLast()	62
5.18.2.9 isItemListEmpty()	64
5.18.2.10 isItemListIndexValid()	64
5.18.2.11 itemListLength()	64
5.18.2.12 newItemList()	65
5.18.2.13 newItemListNode()	65
5.18.2.14 setItem()	65
5.19 item_list.h File Reference	66
5.19.1 Detailed Description	67
5.19.2 Macro Definition Documentation	67
5.19.2.1 next	67
5.19.2.2 value	67
5.19.3 Typedef Documentation	68
5.19.3.1 ItemList	68
5.19.4 Function Documentation	68
5.19.4.1 deleteItemAt()	68
5.19.4.2 deleteItemFirst()	68
5.19.4.3 deleteItemLast()	69
5.19.4.4 getItem()	69

5.19.4.5 indexOfItem()	69
5.19.4.6 insertItemAt()	70
5.19.4.7 insertItemFirst()	70
5.19.4.8 insertItemLast()	70
5.19.4.9 isItemEmpty()	71
5.19.4.10 isItemIndexValid()	71
5.19.4.11 itemListLength()	71
5.19.4.12 newItemList()	72
5.19.4.13 newItemListNode()	72
5.19.4.14 setItem()	72
5.20 item_list.h	73
5.21 item_queue.c File Reference	73
5.21.1 Detailed Description	74
5.21.2 Function Documentation	74
5.21.2.1 dequeue()	74
5.21.2.2 enqueue()	74
5.21.2.3 isEmpty()	74
5.21.2.4 newItemQueue()	75
5.21.2.5 peekHeadTime()	75
5.22 item_queue.h File Reference	75
5.22.1 Detailed Description	76
5.22.2 Macro Definition Documentation	76
5.22.2.1 head	76
5.22.2.2 headIndex	77
5.22.2.3 tail	77
5.22.2.4 tailIndex	77
5.22.3 Function Documentation	77
5.22.3.1 dequeue()	78
5.22.3.2 enqueue()	78
5.22.3.3 isEmpty()	78
5.22.3.4 newItemQueue()	79
5.22.3.5 peekHeadTime()	79
5.23 item_queue.h	79
5.24 item_stack.c File Reference	80
5.24.1 Detailed Description	80
5.24.2 Function Documentation	80
5.24.2.1 _clampCapacity()	80
5.24.2.2 doubleCapacity()	81
5.24.2.3 incrementCapacity()	81
5.24.2.4 isStackEmpty()	81
5.24.2.5 isStackFull()	82
5.24.2.6 newItemStack()	82

5.24.2.7 pop()	82
5.24.2.8 push()	83
5.25 item_stack.h File Reference	83
5.25.1 Detailed Description	84
5.25.2 Macro Definition Documentation	84
5.25.2.1 capacity	84
5.25.2.2 ITEM_STACK_MAX_CAPACITY	84
5.25.2.3 top	84
5.25.2.4 topIndex	85
5.25.3 Function Documentation	85
5.25.3.1 _clampCapacity()	85
5.25.3.2 doubleCapacity()	85
5.25.3.3 incrementCapacity()	86
5.25.3.4 isEmpty()	86
5.25.3.5 isStackFull()	86
5.25.3.6 newItemStack()	87
5.25.3.7 pop()	87
5.25.3.8 push()	87
5.26 item_stack.h	88
5.27 location.c File Reference	88
5.27.1 Detailed Description	89
5.27.2 Function Documentation	89
5.27.2.1 isAt()	89
5.27.2.2 isLocationDefined()	90
5.27.2.3 isLocationIdentical()	90
5.27.2.4 newLocation()	90
5.27.2.5 setAsDropOffPlace()	91
5.27.2.6 setAsPickUpPlace()	91
5.27.2.7 setAsPlayerPlace()	91
5.27.2.8 setAsReachable()	92
5.27.2.9 toggleAsPlayerPlace()	92
5.27.2.10 unsetAsDropOffPlace()	92
5.27.2.11 unsetAsPickUpPlace()	92
5.27.2.12 unsetAsPlayerPlace()	93
5.27.2.13 unsetAsReachable()	93
5.27.2.14 writeLocationSymbol()	93
5.27.3 Variable Documentation	93
5.27.3.1 NULL_LOCATION	94
5.28 location.h File Reference	94
5.28.1 Detailed Description	95
5.28.2 Macro Definition Documentation	95
5.28.2.1 coord	95

5.28.2.2 id	95
5.28.2.3 symbol	96
5.28.3 Function Documentation	96
5.28.3.1 isAt()	96
5.28.3.2 isLocationDefined()	96
5.28.3.3 isLocationIdentical()	97
5.28.3.4 newLocation()	97
5.28.3.5 setAsDropOffPlace()	98
5.28.3.6 setAsPickUpPlace()	98
5.28.3.7 setAsPlayerPlace()	98
5.28.3.8 setAsReachable()	98
5.28.3.9 toggleAsPlayerPlace()	99
5.28.3.10 unsetAsDropOffPlace()	99
5.28.3.11 unsetAsPickUpPlace()	99
5.28.3.12 unsetAsPlayerPlace()	100
5.28.3.13 unsetAsReachable()	100
5.28.3.14 writeLocationSymbol()	100
5.28.4 Variable Documentation	100
5.28.4.1 NULL_LOCATION	100
5.29 location.h	101
5.30 location_list.c File Reference	101
5.30.1 Detailed Description	102
5.30.2 Function Documentation	102
5.30.2.1 _getLocationByCoord()	102
5.30.2.2 _getLocationById()	103
5.30.2.3 _getLocationBySymbol()	103
5.30.2.4 compactList()	104
5.30.2.5 dealocateLocationList()	104
5.30.2.6 deleteLast()	104
5.30.2.7 growList()	104
5.30.2.8 insertLast()	105
5.30.2.9 isIndexEff()	105
5.30.2.10 isIndexValid()	105
5.30.2.11 isLocationListEmpty()	106
5.30.2.12 isLocationListFull()	106
5.30.2.13 length()	106
5.30.2.14 newLocationList()	107
5.30.2.15 shrinkList()	107
5.30.2.16 sortLocationListByCoord()	107
5.31 location_list.h File Reference	108
5.31.1 Detailed Description	109
5.31.2 Macro Definition Documentation	109

5.31.2.1 buffer	109
5.31.2.2 capacity	109
5.31.2.3 IElem	110
5.31.2.4 neff	110
5.31.3 Function Documentation	110
5.31.3.1 _getLocationByCoord()	110
5.31.3.2 _getLocationById()	111
5.31.3.3 _getLocationBySymbol()	111
5.31.3.4 compactList()	112
5.31.3.5 deallocateLocationList()	112
5.31.3.6 deleteLast()	112
5.31.3.7 growList()	112
5.31.3.8 insertLast()	113
5.31.3.9 isIndexEff()	113
5.31.3.10 isIndexValid()	113
5.31.3.11 isLocationListEmpty()	114
5.31.3.12 isLocationListFull()	114
5.31.3.13 length()	114
5.31.3.14 newLocationList()	115
5.31.3.15 shrinkList()	115
5.31.3.16 sortLocationListByCoord()	115
5.32 location_list.h	116
5.33 location_matrix.c File Reference	116
5.33.1 Detailed Description	117
5.33.2 Function Documentation	117
5.33.2.1 ISetElem()	117
5.33.2.2 newLocationMatrix()	117
5.34 location_matrix.h File Reference	118
5.34.1 Detailed Description	118
5.34.2 Macro Definition Documentation	118
5.34.2.1 cols	118
5.34.2.2 elem	119
5.34.2.3 rows	119
5.34.3 Function Documentation	119
5.34.3.1 ISetElem()	119
5.34.3.2 newLocationMatrix()	120
5.35 location_matrix.h	120
5.36 point.c File Reference	120
5.36.1 Detailed Description	121
5.36.2 Function Documentation	121
5.36.2.1 displayPoint()	121
5.36.2.2 isPointBefore()	121

5.36.2.3 isPointIdentical()	122
5.36.2.4 newPoint()	122
5.37 point.h File Reference	123
5.37.1 Detailed Description	123
5.37.2 Macro Definition Documentation	123
5.37.2.1 abs	123
5.37.2.2 ord	124
5.37.3 Function Documentation	124
5.37.3.1 displayPoint()	124
5.37.3.2 isPointBefore()	124
5.37.3.3 isPointIdentical()	125
5.37.3.4 newPoint()	125
5.38 point.h	125
5.39 state.h File Reference	126
5.39.1 Detailed Description	126
5.39.2 Function Documentation	126
5.39.2.1 moveItemToProgressList()	126
5.39.2.2 newState()	127
5.39.2.3 reevaluate()	127
5.40 state.h	128
Index	129

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:

BooleanMatrix	Matriks bernilai boolean	7
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 run-time)	8
GadgetList	List statik berisi tepat 5 Gadget	9
GameMap	Tipe data berisi ukuran map, matriks adjacency, list lokasi, dan matriks lokasi	10
Item	Struktur tipe data Item dan pesanan	11
ItemListNode	Node dari tipe data linked list Item	12
ItemQueue	Antrian Item terurut berdasarkan waktu pesanan masuk	13
ItemStack	Tumpukan Item pada tas	14
Location	Struktur tipe data lokasi yang memuat koordinat, simbol, dan id	15
LocationList	List dinamis berisi data Location	17
LocationMatrix	Matriks berisi data Location	18
Point	Struktur tipe data titik	19
State	Game state & life cycle	20

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

boolean.h	Definisi tipe data boolean	23
boolean_matrix.c	Implementasi tipe data BooleanMatrix	24
boolean_matrix.h	Header file untuk tipe data BooleanMatrix	25
gadget.c	Implementasi tipe data Gadget	28
gadget.h	Header file untuk tipe data Gadget	30
gadget_list.c	Implementasi tipe data GadgetList . Tipe data ini digunakan untuk inventory pada game	34
gadget_list.h	Header file untuk tipe data GadgetList	37
game_map.c	Implementasi tipe data GameMap	40
game_map.h	Header file untuk tipe data GameMap	43
item.c	Implementasi tipe data Item . Digunakan untuk mencatat order, termasuk Todo list dan In Progress list	49
item.h	Header file untuk tipe data Item	52
item_list.c	Implementasi tipe data ItemList . Digunakan untuk Todo List dan In Progress List	59
item_list.h	Header file untuk tipe data ItemList	66
item_queue.c	Implementasi tipe data ItemQueue . Hanya digunakan untuk antrian pesanan masuk	73
item_queue.h	Header file untuk tipe data ItemQueue	75
item_stack.c	Implementasi tipe data ItemStack . Hanya digunakan untuk INVENTORY	80
item_stack.h	Header file untuk tipe data ItemStack	83

location.c	Implementasi tipe data Location . Digunakan untuk merepresentasikan lokasi pada GameMap .	88
location.h	Header file untuk tipe data Location	94
location_list.c	Implementasi tipe data LocationList . Digunakan menyimpan daftar lokasi yang ada, dan daftar lokasi yang adjacent dengan suatu lokasi	101
location_list.h	Header file untuk tipe data LocationList	108
location_matrix.c	Implementasi tipe data LocationMatrix	116
location_matrix.h	Header file untuk tipe data LocationMatrix	118
point.c	Implementasi tipe data Point . Digunakan untuk merepresentasikan sebuah koordinat lokasi . .	120
point.h	Header file untuk tipe data Point	123
state.h	Header file untuk State game	126

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](#) teratur 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

<i>rows</i>	Banyak baris efektif.
<i>cols</i>	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

<i>b</i>	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

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

5.4.2.3 rows

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

Mengembalikan banyak baris efektif [BooleanMatrix](#) *b*.

Parameters

<i>b</i>	BooleanMatrix instance.
----------	---

5.4.3 Function Documentation

5.4.3.1 newBooleanMatrix()

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

Constructor untuk membuat [BooleanMatrix](#) baru.

Parameters

<i>rows</i>	Banyak baris efektif (rowEff).
<i>cols</i>	Banyak kolom efektif (colEff).

Returns

[BooleanMatrix](#) instance.

Parameters

<i>rows</i>	Banyak baris efektif.
<i>cols</i>	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](#) [SENER_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](#) [SENER_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

<i>gadget1</i>	Gadget instance.
<i>gadget2</i>	Gadget instance.

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

<i>g</i>	Gadget instance.
----------	----------------------------------

5.7.2.2 name

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

Mengambil property name dari sebuah [Gadget](#).

Parameters

<i>g</i>	Gadget instance.
----------	----------------------------------

5.7.2.3 price

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

Mengambil property price dari sebuah [Gadget](#).

Parameters

<i>g</i>	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

<i>gadget1</i>	Gadget instance.
<i>gadget2</i>	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

<i>gList</i>	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

<i>gList</i>	GadgetList instance.
<i>index</i>	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

<i>gList</i>	GadgetList instance.
--------------	--------------------------------------

Returns

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

5.9.2.4 isGadgetListFull()

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

Mengecek apakah gadget *gList* penuh atau tidak.

Parameters

<i>gList</i>	GadgetList instance.
--------------	--------------------------------------

Returns

true jika semua elemen *gList* bukanlah gadget yang tidak terdefinisi, 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

<i>gList</i>	GadgetList instance.
<i>index</i>	Indeks gList yang akan di-set.
<i>g</i>	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

<i>gList</i>	GadgetList instance.
--------------	--------------------------------------

5.10.2.2 `getGadget()`

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

Mengambil [Gadget](#) instance dari gList pada indeks index.

Parameters

<i>gList</i>	GadgetList instance.
<i>index</i>	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

<i>gList</i>	GadgetList instance.
<i>index</i>	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

<i>gList</i>	GadgetList instance.
--------------	--------------------------------------

Returns

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

5.10.2.4 isGadgetListFull()

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

Mengecek apakah gadget gList penuh atau tidak.

Parameters

<i>gList</i>	GadgetList instance.
--------------	----------------------

Returns

true jika semua elemen gList bukanlah gadget yang tidak terdefinisi, 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

<i>gList</i>	GadgetList instance.
<i>index</i>	Indeks gList yang akan di-set.
<i>g</i>	Gadget instance.

5.11 gadget_list.h

[Go to the documentation of this file.](#)

```

1
2
3
4
5
6 #ifndef GADGET_LIST_H
7 #define GADGET_LIST_H
8
9 #include "boolean.h"
10 #include "gadget.h"
11
12
13
14
15
16 typedef struct
17 {
18     Gadget contents[5];
19 } GadgetList;
20
21
22
23
24
25
26
27
28
29 GadgetList newGadgetList();
30
31
32
33
34
35
36
37
38 boolean isGadgetListEmpty(GadgetList gList);
39
40
41
42
43
44
45
46
47 boolean isGadgetListFull(GadgetList gList);
48
49
50
51
52
53
54
55
56 Gadget getGadget(GadgetList gList, int index);
57
58
59
60
61
62
63
64
65 void setGadget(GadgetList *gList, int index, Gadget g);
66
67
68
69
70
71
72
73 void displayGadget(GadgetList gList);
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637

```

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

<i>m</i>	GameMap instance.
<i>currentLocation</i>	Lokasi saat ini.

5.12.2.3 `displayGameMap()`

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

Menampilkan map game ke console output.

Parameters

<i>m</i>	GameMap instance yang akan ditampilkan.
----------	---

5.12.2.4 getLocationByCoord()

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

Mengambil [Location](#) instance berdasarkan koordinat.

Parameters

<i>m</i>	GameMap instance.
<i>p</i>	Koordinat Location .

Returns

[Location](#) dengan koordinat p.

5.12.2.5 getLocationById()

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

Mengambil [Location](#) instance berdasarkan id.

Parameters

<i>m</i>	GameMap instance.
<i>id</i>	Id Location .

Returns

[Location](#) dengan id 'id'.

5.12.2.6 getLocationBySymbol()

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

Mengambil [Location](#) instance berdasarkan simbol.

Parameters

<i>m</i>	GameMap instance
<i>symbol</i>	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

<i>a</i>	Location pertama.
<i>b</i>	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

<i>hSize</i>	Panjang map (horizontal).
<i>vSize</i>	Lebar map (vertikal).
<i>adjMatrix</i>	Matriks adjacency dari lokasi-lokasi yang ada.
<i>locations</i>	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

<i>m</i>	GameMap instance
----------	----------------------------------

5.13.2.2 locList

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

Mengembalikan list lokasi yang ada pada *m*.

See also

[LocationList](#)

Parameters

<i>m</i>	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

<i>m</i>	GameMap instance
----------	----------------------------------

5.13.2.4 mapLength

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

Mengembalikan panjang [GameMap](#) *m*.

Parameters

<i>m</i>	GameMap instance.
----------	-----------------------------------

5.13.2.5 mapWidth

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

Mengembalikan lebar [GameMap](#) *m*.

Parameters

<i>m</i>	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

<i>m</i>	GameMap instance.
<i>currentLocation</i>	Lokasi saat ini.

5.13.3.2 displayGameMap()

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

Menampilkan map game ke console output.

Parameters

<i>m</i>	GameMap instance yang akan ditampilkan.
----------	---

5.13.3.3 getLocationByCoord()

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

Mengambil Location instance berdasarkan koordinat.

Parameters

<i>m</i>	GameMap instance.
<i>p</i>	Koordinat Location.

Returns

Location dengan koordinat p.

5.13.3.4 getLocationById()

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

Mengambil Location instance berdasarkan id.

Parameters

<i>m</i>	GameMap instance.
<i>id</i>	Id Location.

Returns

Location dengan id 'id'.

5.13.3.5 getLocationBySymbol()

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

Mengambil [Location](#) instance berdasarkan simbol.

Parameters

<i>m</i>	GameMap instance
<i>symbol</i>	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

<i>a</i>	Location pertama.
<i>b</i>	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

<i>hSize</i>	Panjang map (horizontal).
<i>vSize</i>	Lebar map (vertikal).
<i>adjMatrix</i>	Matriks adjacency dari lokasi-lokasi yang ada.
<i>locations</i>	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

<i>item</i>	<code>Item</code> 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

<i>item1</i>	Item instance.
<i>item2</i>	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

<i>item</i>	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

<i>item</i>	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

<i>item</i>	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

<i>orderTime</i>	Waktu order item.
<i>pickupLocation</i>	Lokasi pick up item.
<i>dropOffLocation</i>	Lokasi drop off item.
<i>type</i>	Tipe item.
<i>perishTime</i>	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

<i>item</i>	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

<i>item</i>	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

<i>item</i>	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

<i>item</i>	Item instance.
-------------	--------------------------------

5.16.2.8 pickUpLoc

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

Mengambil lokasi pick up item.

Parameters

<i>item</i>	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

<i>item</i>	<code>Item</code> 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

<i>item1</i>	Item instance.
<i>item2</i>	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

<i>item</i>	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

<i>item</i>	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

<i>item</i>	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

<i>orderTime</i>	Waktu order item.
<i>pickupLocation</i>	Lokasi pick up item.
<i>dropOffLocation</i>	Lokasi drop off item.
<i>type</i>	Tipe item.
<i>perishTime</i>	Waktu hangus item.

Returns

Item instance baru.

5.17 item.h

[Go to the documentation of this file.](#)


```

1
2
3
4
5
6 #ifndef ITEM_H
7 #define ITEM_H
8
9 #include "location.h"
10
11
12
13
14 typedef int ItemType;
15
16 #define NORMAL 0
17
18 #define HEAVY 1
19
20 #define PERISHABLE 2
21
22 #define VIP 3
23
24
25
26
27
28
29
30
31
32
33 #define UNTIMED -1
34
35
36 typedef struct
37 {
38     int orderTime;
39     Location pickUpLocation;
40     Location dropOffLocation;
41     ItemType type;
42     int perishTime;
43 } Item;
44
45 #define pickUpLoc(item) (item).pickUpLocation
46 #define dropOffLoc(item) (item).dropOffLocation
47 #define itemType(item) (item).type
48 #define perishTime(item) (item).perishTime
49 #define orderTime(item) (item).orderTime
50
51 Item newItem(int orderTime, Location pickUpLocation, Location dropOffLocation, ItemType type, int
    perishTime);
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111 boolean isItemIdentical(Item item1, Item item2);
112
113
114
115
116
117
118
119
120 boolean isNormalItem(Item item);
121
122
123
124
125
126
127
128
129 boolean isHeavyItem(Item item);
130
131
132
133
134
135
136
137
138 boolean isPerishableItem(Item item);
139
140
141
142
143
144
145
146
147 boolean isVIPItem(Item item);
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249 #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 newList \(\)](#)
Constructor untuk membuat ItemList baru.
- [ItemListNode newListNode \(Item item\)](#)
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 & menghapus 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

	<i>iList</i>	ItemList yang akan dilakukan penghapusan.
	<i>index</i>	Indeks Item yang akan dihapus.
out	<i>item</i>	Item pada indeks index.

5.18.2.2 deleteItemFirst()

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

Mengambil & menghapus [Item](#) pertama pada iList.

Parameters

	<i>iList</i>	ItemLIst yang akan dihapus nilai pertamanya.
out	<i>item</i>	Item di posisi pertama iList.

5.18.2.3 deleteItemLast()

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

Mengambil & menghapus [Item](#) terakhir iList.

Parameters

<i>iList</i>	ItemLIst instance.
<i>item</i>	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

<i>iList</i>	ItemLIst instance.
<i>index</i>	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

<i>iList</i>	ItemList instance.
<i>item</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	Indeks yang akan dimasukkan Item.
<i>item</i>	Item instance.

5.18.2.7 insertItemFirst()

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

Insert Item di awal list iList.

Parameters

<i>iList</i>	ItemList instance.
<i>item</i>	Item instance.

5.18.2.8 insertItemLast()

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

Insert [Item](#) di akhir list iList.

Parameters

<i>iList</i>	ItemList instance.
<i>item</i>	Item instance.

5.18.2.9 isItemListEmpty()

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

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

<i>iList</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	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

<i>iList</i>	Item <code>List</code> instance.
--------------	----------------------------------

Returns

Panjang dari `iList`.

5.18.2.12 newItemList()

```
ItemList newItemList ( )
```

Constructor untuk membuat `ItemList` baru.

Returns

`ItemList` kosong.

5.18.2.13 newListNode()

```
ItemListNode newListNode (
    Item item )
```

Constructor untuk membuat `ItemListNode` baru.

Parameters

<i>item</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	Indeks iList yang akan di-set.
<i>item</i>	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 & menghapus 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

<i>node</i>	ItemListNode instance.
-------------	------------------------

5.19.2.2 value

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

Mengambil value dari sebuah ItemListNode.

Parameters

<i>node</i>	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

	<i>iList</i>	ItemList yang akan dilakukan penghapusan.
	<i>index</i>	Indeks Item yang akan dihapus.
out	<i>item</i>	Item pada indeks index.

5.19.4.2 deleteItemFirst()

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

Mengambil & menghapus [Item](#) pertama pada iList.

Parameters

	<i>iList</i>	ItemList yang akan dihapus nilai pertamanya.
out	<i>item</i>	Item di posisi pertama iList.

5.19.4.3 deleteItemLast()

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

Mengambil & menghapus **Item** terakhir iList.

Parameters

<i>iList</i>	ItemList instance.
<i>item</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	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

<i>iList</i>	ItemList instance.
<i>item</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	Indeks yang akan dimasukkan Item .
<i>item</i>	Item instance.

5.19.4.7 insertItemFirst()

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

Insert [Item](#) di awal list iList.

Parameters

<i>iList</i>	ItemList instance.
<i>item</i>	Item instance.

5.19.4.8 insertItemLast()

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

Insert [Item](#) di akhir list iList.

Parameters

<i>iList</i>	ItemList instance.
<i>item</i>	Item instance.

5.19.4.9 isItemListEmpty()

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

Mengecek apakah suatu ItemList kosong atau tidak.

Parameters

<i>iList</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	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

<i>iList</i>	ItemList instance.
--------------	--------------------

Returns

Panjang dari iList.

5.19.4.12 newList()

```
ItemList newList ( )
```

Constructor untuk membuat ItemList baru.

Returns

ItemList kosong.

5.19.4.13 newListNode()

```
ItemListNode newListNode (
    Item item )
```

Constructor untuk membuat ItemListNode baru.

Parameters

<i>item</i>	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

<i>iList</i>	ItemList instance.
<i>index</i>	Indeks iList yang akan di-set.
<i>item</i>	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 newList();
50
57 ItemListNode newListNode(Item item);
58
65 boolean isEmpty(ItemList iList);
66
75 boolean isValidIndex(ItemList iList, int index);
76
83 int listLength(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 newListQueue \(\)](#)
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

<code>q</code>	<code>ItemQueue</code> instance.
<code>item</code>	<code>Item</code> terdepan pada antrian <code>q</code> .

5.21.2.2 `enqueue()`

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

Menambah `Item` item pada antrian `q`.

Parameters

<code>q</code>	<code>ItemQueue</code> instance.
<code>item</code>	<code>Item</code> instance.

5.21.2.3 `isEmpty()`

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

Mengecek apakah queue `q` kosong atau tidak.

Parameters

<i>q</i>	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

<i>q</i>	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

<i>q</i>	ItemQueue instance.
----------	-------------------------------------

5.22.2.2 headIndex

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

Mengambil indeks head pada antrian *q*.

Parameters

<i>q</i>	ItemQueue instance.
----------	-------------------------------------

5.22.2.3 tail

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

Mengambil tail [Item](#) pada antrian *q*.

Parameters

<i>q</i>	ItemQueue instance.
----------	-------------------------------------

5.22.2.4 tailIndex

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

Mengambil indeks tail pada antrian *q*.

Parameters

<i>q</i>	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

<i>q</i>	ItemQueue instance.
<i>item</i>	Item terdepan pada antrian q.

5.22.3.2 enqueue()

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

Menambah **Item** item pada antrian q.

Parameters

<i>q</i>	ItemQueue instance.
<i>item</i>	Item instance.

5.22.3.3 isEmpty()

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

Mengecek apakah queue q kosong atau tidak.

Parameters

<i>q</i>	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

<code>q</code>	<code>ItemQueue</code> instance.
----------------	----------------------------------

Returns

Nilai pesanan `Item` terdepan q.

5.23 item_queue.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef ITEM_QUEUE_H
3 #define ITEM_QUEUE_H
4
5 #include "boolean.h"
6 #include "item.h"
7
8 typedef struct
9 {
10     int headIndex;
11     int tailIndex;
12     Item buffer[30];
13 } ItemQueue;
14
15 #define headIndex(q) (q).headIndex
16 #define tailIndex(q) (q).tailIndex
17 #define head(q) (q).buffer[(q).headIndex]
18 #define tail(q) (q).buffer[(q).tailIndex]
19
20 ItemQueue newItemQueue();
21
22 int peekHeadTime(ItemQueue q);
23
24 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 isEmpty](#) ([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

<i>stack</i>	ItemStack instance.
--------------	-------------------------------------

5.24.2.2 doubleCapacity()

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

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

Parameters

<i>stack</i>	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

<i>stack</i>	ItemStack instance.
--------------	-------------------------------------

5.24.2.4 isStackEmpty()

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

Mengecek apakah stack kosong atau tidak.

Parameters

<i>stack</i>	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

<i>stack</i>	<code>ItemStack</code> instance.
--------------	----------------------------------

Returns

true jika stack penuh, false selainnya.

5.24.2.6 newItemStack()

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

Constructor untuk membuat `ItemStack` baru.

Parameters

<i>capacity</i>	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

	<i>stack</i>	<code>ItemStack</code> instance.
out	<i>item</i>	<code>Item</code> yang diambil dari atas stack.

5.24.2.8 push()

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

Memasukkan item ke atas stack.

Parameters

<i>stack</i>	ItemStack instance.
<i>item</i>	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 isEmpty (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	<code>ItemStack</code> 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

<i>s</i>	ItemStack instance.
----------	-------------------------------------

5.25.2.4 topIndex

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

Mengambil indeks teratas stack *s*.

Parameters

<i>s</i>	ItemStack instance.
----------	-------------------------------------

5.25.3 Function Documentation

5.25.3.1 _clampCapacity()

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

Membatasi kapasitas stack jika kapasitas melebihi batas.

Parameters

<i>stack</i>	ItemStack instance.
--------------	-------------------------------------

5.25.3.2 doubleCapacity()

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

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

Parameters

<i>stack</i>	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

<i>stack</i>	ItemStack instance.
--------------	-------------------------------------

5.25.3.4 isStackEmpty()

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

Mengecek apakah stack kosong atau tidak.

Parameters

<i>stack</i>	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

<i>stack</i>	ItemStack instance.
--------------	-------------------------------------

Returns

true jika stack penuh, false selainnya.

5.25.3.6 newItemStack()

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

Constructor untuk membuat [ItemStack](#) baru.

Parameters

<i>capacity</i>	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

	<i>stack</i>	ItemStack instance.
out	<i>item</i>	Item yang diambil dari atas stack.

5.25.3.8 push()

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

Memasukkan item ke atas stack.

Parameters

<i>stack</i>	ItemStack instance.
<i>item</i>	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 isEmpty(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](#) *)
- *Unset lokasi sebagai tempat pick up [Item](#).*
void [setAsDropOffPlace](#) ([Location](#) *)
- *Set lokasi sebagai tempat drop off [Item](#).*
void [unsetAsDropOffPlace](#) ([Location](#) *)
- *Unset lokasi sebagai tempat drop off [Item](#).*
void [setAsReachable](#) ([Location](#) *)
- *Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.*
void [unsetAsReachable](#) ([Location](#) *)
- *Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.*
void [setAsPlayerPlace](#) ([Location](#) *)
- *Set lokasi sebagai lokasi player.*
void [unsetAsPlayerPlace](#) ([Location](#) *)
- *Unset lokasi sebagai lokasi player.*
void [toggleAsPlayerPlace](#) ([Location](#) *)
- *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

<i>l</i>	Location instance.
<i>p</i>	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

/	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

/1	Location instance.
/2	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

<i>id</i>	Identifier lokasi.
<i>symbol</i>	Simbol lokasi.
<i>coordinate</i>	Koordinat lokasi.

Returns

[Location](#) instance baru.

5.27.2.5 setAsDropOffPlace()

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

Set lokasi sebagai tempat drop off [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.27.2.6 setAsPickUpPlace()

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

Set lokasi sebagai tempat pick up [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.27.2.7 setAsPlayerPlace()

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

Set lokasi sebagai lokasi player.

Parameters

/	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

/	Location instance.
---	--------------------

5.27.2.9 toggleAsPlayerPlace()

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

Toggle lokasi sebagai lokasi player.

Parameters

/	Location instance.
---	--------------------

5.27.2.10 unsetAsDropOffPlace()

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

Unset lokasi sebagai tempat drop off [Item](#).

Parameters

/	Location instance.
---	--------------------

5.27.2.11 unsetAsPickUpPlace()

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

Unset lokasi sebagai tempat pick up [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.27.2.12 unsetAsPlayerPlace()

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

Unset lokasi sebagai lokasi player.

Parameters

/	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

/	Location instance.
---	------------------------------------

5.27.2.14 writeLocationSymbol()

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

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

Parameters

/	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](#) *)
Set lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void [unsetAsReachable](#) ([Location](#) *)
Unset lokasi sebagai tempat yang dapat dituju relatif dengan lokasi player saat ini.
- void [setAsPlayerPlace](#) ([Location](#) *)
Set lokasi sebagai lokasi player.
- void [unsetAsPlayerPlace](#) ([Location](#) *)
Unset lokasi sebagai lokasi player.
- void [toggleAsPlayerPlace](#) ([Location](#) *)
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

/	Location instance.
---	------------------------------------

5.28.2.2 id

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

Mengambil id lokasi.

Parameters

/	Location instance.
---	------------------------------------

5.28.2.3 symbol

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

Mengambil simbol lokasi.

Parameters

/	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

/	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

<i>l</i>	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

<i>l1</i>	Location instance.
<i>l2</i>	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

<i>id</i>	Identifier lokasi.
<i>symbol</i>	Simbol lokasi.
<i>coordinate</i>	Koordinat lokasi.

Returns

[Location](#) instance baru.

5.28.3.5 setAsDropOffPlace()

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

Set lokasi sebagai tempat drop off [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.28.3.6 setAsPickUpPlace()

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

Set lokasi sebagai tempat pick up [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.28.3.7 setAsPlayerPlace()

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

Set lokasi sebagai lokasi player.

Parameters

/	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

/	Location instance.
---	------------------------------------

5.28.3.9 toggleAsPlayerPlace()

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

Toggle lokasi sebagai lokasi player.

Parameters

/	Location instance.
---	------------------------------------

5.28.3.10 unsetAsDropOffPlace()

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

Unset lokasi sebagai tempat drop off [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.28.3.11 unsetAsPickUpPlace()

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

Unset lokasi sebagai tempat pick up [Item](#).

Parameters

/	Location instance.
---	------------------------------------

5.28.3.12 unsetAsPlayerPlace()

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

Unset lokasi sebagai lokasi player.

Parameters

/	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

/	Location instance.
---	--------------------

5.28.3.14 writeLocationSymbol()

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

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

Parameters

/	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 newListLocationList (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

<i>l</i>	LocationList instance.
<i>p</i>	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

<i>l</i>	LocationList instance.
<i>id</i>	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

<i>l</i>	LocationList instance.
<i>symbol</i>	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

/	LocationList instance.
---	------------------------

5.30.2.5 dealocateLocationList()

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

Menghapus memory list dari heap memory.

Parameters

/	LocationList instance.
---	------------------------

5.30.2.6 deleteLast()

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

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	/	LocaitonList instance.
out	location	Location instance.

5.30.2.7 growList()

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

Menambah kapasitas list l sebanyak num.

Parameters

<i>l</i>	LocationList instance.
<i>num</i>	Banyak kapasitas yang akan ditambah.

5.30.2.8 insertLast()

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

Memasukkan [Location](#) ke akhir list.

Parameters

<i>l</i>	LocationList instance.
<i>location</i>	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

<i>l</i>	LocationList instance.
<i>i</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

<i>/</i>	LocationList instance.
<i>i</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

<i>/</i>	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

<i>/</i>	LocationList instance.
----------	--

Returns

true jika list l penuh, false selainnya.

5.30.2.13 length()

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


Mengembalikan panjang list l.

Parameters

<i>l</i>	LocationList instance.
----------	--

Returns

Panjang list l.

5.30.2.14 newLocationList()

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

Constructor untuk membuat [LocationList](#) baru.

Parameters

<i>capacity</i>	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

<i>l</i>	LocationList instance.
<i>num</i>	Banyak kapasitas yang akan dikurangi.

5.30.2.16 sortLocationListByCoord()

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

Mengurutkan [Location](#) dalam *l* berdasarkan koordinat.

See also

[Location](#)

Parameters

/	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 [IElem](#)(*l*, *i*) (*l*).[buffer](#)[*i*]
Mengambil elemen list pada indeks tertentu.

Functions

- [LocationList](#) [newLocationList](#) (int [capacity](#))
Constructor untuk membuat [LocationList](#) baru.
- void [deallocateLocationList](#) ([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

/	LocationList instance.
---	--

5.31.2.2 capacity

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

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

Mengambil kapasitas elemen list.

Parameters

/	LocationList instance.
---	--

5.31.2.3 IElem

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

Mengambil elemen list pada indeks tertentu.

Parameters

/	LocationList instance.
i	Indeks elemen yang akan diambil.

5.31.2.4 neff

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

Mengambil banyak elemen list.

Parameters

/	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

<i>l</i>	LocationList instance.
<i>p</i>	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

<i>l</i>	LocationList instance.
<i>id</i>	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

<i>l</i>	LocationList instance.
<i>symbol</i>	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

/	LocationList instance.
---	------------------------

5.31.3.5 dealocateLocationList()

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

Menghapus memory list dari heap memory.

Parameters

/	LocationList instance.
---	------------------------

5.31.3.6 deleteLast()

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

Menghapus & mengambil Location terakhir di dalam list.

Parameters

	/	LocaitonList instance.
out	location	Location instance.

5.31.3.7 growList()

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

Menambah kapasitas list l sebanyak num.

Parameters

<i>l</i>	LocationList instance.
<i>num</i>	Banyak kapasitas yang akan ditambah.

5.31.3.8 insertLast()

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

Memasukkan [Location](#) ke akhir list.

Parameters

<i>l</i>	LocationList instance.
<i>location</i>	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

<i>l</i>	LocationList instance.
<i>i</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

<i>/</i>	LocationList instance.
<i>i</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

<i>/</i>	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

<i>/</i>	LocationList instance.
----------	--

Returns

true jika list l penuh, false selainnya.

5.31.3.13 length()

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


Mengembalikan panjang list l.

Parameters

<i>l</i>	LocationList instance.
----------	--

Returns

Panjang list l.

5.31.3.14 newLocationList()

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

Constructor untuk membuat [LocationList](#) baru.

Parameters

<i>capacity</i>	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

<i>l</i>	LocationList instance.
<i>num</i>	Banyak kapasitas yang akan dikurangi.

5.31.3.16 sortLocationListByCoord()

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

Mengurutkan [Location](#) dalam l berdasarkan koordinat.

See also

[Location](#)

Parameters

/	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
13 typedef struct
14 {
15     Location *buffer;
16     int nEff;
17     int capacity;
18 } LocationList;
19
20 #define neff(l) (l).nEff
21 #define capacity(l) (l).capacity
22 #define buffer(l) (l).buffer
23 #define lElem(l, i) (l).buffer[i]
24
25 LocationList newLocationList(int capacity);
26 void dealocateLocationList(LocationList *l);
27 int length(LocationList l);
28
29 boolean isIndexValid(LocationList l, int i);
30 boolean isIndexEff(LocationList l, int i);
31 boolean isLocationListEmpty(LocationList l);
32 boolean isLocationListFull(LocationList l);
33 void insertLast(LocationList *l, Location location);
34 void deleteLast(LocationList *l, Location *location);
35 void growList(LocationList *l, int num);
36 void shrinkList(LocationList *l, int num);
37 void compactList(LocationList *l);
38 void sortLocationListByCoord(LocationList *l);
39
40 Location _getLocationById(LocationList l, int id);
41 Location _getLocationBySymbol(LocationList l, char symbol);
42 Location _getLocationByCoord(LocationList l, Point p);
43 #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 `ISetElem` ([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 ISetElem()

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

Set elemen [LocationMatrix](#) `l` pada indeks tertentu.

Parameters

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

5.33.2.2 newLocationMatrix()

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

Constructor untuk membuat [LocationMatrix](#) baru.

Parameters

<i>rows</i>	Banyak baris matriks.
<i>cols</i>	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 [ISetElem](#) ([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

<i>l</i>	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

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

5.34.2.3 rows

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

Mengambil banyak baris matriks *l*.

Parameters

<i>l</i>	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

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

5.34.3.2 newLocationMatrix()

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

Constructor untuk membuat [LocationMatrix](#) baru.

Parameters

<i>rows</i>	Banyak baris matriks.
<i>cols</i>	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

<i>p</i>	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

<i>p1</i>	Point instance.
<i>p2</i>	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

<i>p1</i>	Point instance.
<i>p2</i>	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

<i>x</i>	Absis.
<i>y</i>	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

<i>p</i>	Point instance.
----------	---------------------------------

5.37.2.2 ord

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

Mengambil ordinat dari suatu titik.

Parameters

<i>p</i>	Point instance.
----------	---------------------------------

5.37.3 Function Documentation**5.37.3.1 displayPoint()**

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

Menuliskan titik p ke console output.

Parameters

<i>p</i>	Point instance.
----------	---------------------------------

5.37.3.2 isPointBefore()

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

Mengecek apakah suatu titik berada pada koordinat "sebelum" titik lainnya, yaitu $x_1 < x_2$ atau $x_1 = x_2$ & $y_1 < y_2$.

Parameters

<i>p1</i>	Point instance.
<i>p2</i>	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

<i>p1</i>	Point instance.
<i>p2</i>	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

<i>x</i>	Absis.
<i>y</i>	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

<i>state</i>	State saat ini.
<i>indexTodo</i>	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

<i>m</i>	GameMap instance dari game instance ini.
<i>todo</i>	Todo List dari game instance ini.
<i>inProgress</i>	In Progress List dari game instance ini.
<i>bag</i>	Tas player pada game instance ini.
<i>order</i>	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

<i>state</i>	State saat ini.
--------------	---------------------------------

5.40 state.h

[Go to the documentation of this file.](#)

```
1
2
3
4
5
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
15
16
17
18 typedef struct
19 {
20     GameMap gameMap;
21     ItemList todoList;
22     ItemList inProgressList;
23     ItemStack bag;
24     ItemQueue order;
25     GadgetList inventory;
26     int cash;
27     Location currentLocation;
28 } State;
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71 State newState(GameMap m, ItemList todo, ItemList inProgress, ItemStack bag, ItemQueue order);
72
73
74
75
76
77
78
79
80
81
82 void moveItemToProgressList(State *state, int indexTodo);
83
84
85
86
87
88
89
90 void reevaluate(State *state);
91
92 #endif
```

Index

- `_adjacency`
 - `GameMap`, [10](#)
 - `_clampCapacity`
 - `item_stack.c`, [80](#)
 - `item_stack.h`, [85](#)
 - `_getAdjacentLocations`
 - `game_map.c`, [41](#)
 - `_getLocationByCoord`
 - `location_list.c`, [102](#)
 - `location_list.h`, [110](#)
 - `_getLocationById`
 - `location_list.c`, [103](#)
 - `location_list.h`, [111](#)
 - `_getLocationBySymbol`
 - `location_list.c`, [103](#)
 - `location_list.h`, [111](#)
 - `_locationMatrix`
 - `GameMap`, [10](#)
 - `_locations`
 - `GameMap`, [10](#)
- `abs`
 - `point.h`, [123](#)
- `adjMatrix`
 - `game_map.h`, [44](#)
- `bag`
 - `State`, [21](#)
- `boolean`
 - `boolean.h`, [23](#)
- `boolean.h`, [23](#)
 - `boolean`, [23](#)
 - `false`, [23](#)
 - `true`, [24](#)
- `boolean_matrix.c`, [24](#)
 - `newBooleanMatrix`, [25](#)
- `boolean_matrix.h`, [25](#)
 - `cols`, [26](#)
 - `elem`, [26](#)
 - `newBooleanMatrix`, [27](#)
 - `rows`, [26](#)
- `BooleanMatrix`, [7](#)
 - `colEff`, [7](#)
 - `contents`, [7](#)
 - `rowEff`, [8](#)
- `buffer`
 - `ItemQueue`, [14](#)
 - `ItemStack`, [15](#)
 - `location_list.h`, [109](#)
 - `LocationList`, [18](#)
- `capacity`
 - `item_stack.h`, [84](#)
 - `ItemStack`, [15](#)
 - `location_list.h`, [109](#)
 - `LocationList`, [18](#)
- `cash`
 - `State`, [21](#)
- `colEff`
 - `BooleanMatrix`, [7](#)
 - `LocationMatrix`, [19](#)
- `cols`
 - `boolean_matrix.h`, [26](#)
 - `location_matrix.h`, [118](#)
- `compactList`
 - `location_list.c`, [103](#)
 - `location_list.h`, [111](#)
- `contents`
 - `BooleanMatrix`, [7](#)
 - `GadgetList`, [9](#)
 - `LocationMatrix`, [19](#)
- `coord`
 - `location.h`, [95](#)
- `coordinate`
 - `Location`, [16](#)
- `currentLocation`
 - `State`, [21](#)
- `deallocateLocationList`
 - `location_list.c`, [104](#)
 - `location_list.h`, [112](#)
- `deleteItemAt`
 - `item_list.c`, [60](#)
 - `item_list.h`, [68](#)
- `deleteItemFirst`
 - `item_list.c`, [60](#)
 - `item_list.h`, [68](#)
- `deleteItemLast`
 - `item_list.c`, [61](#)
 - `item_list.h`, [68](#)
- `deleteLast`
 - `location_list.c`, [104](#)
 - `location_list.h`, [112](#)
- `dequeue`
 - `item_queue.c`, [74](#)
 - `item_queue.h`, [77](#)
- `displayAdjacentLocation`
 - `game_map.c`, [41](#)
 - `game_map.h`, [46](#)
- `displayGadget`
 - `gadget_list.c`, [35](#)

- gadget_list.h, [37](#)
- displayGameMap
 - game_map.c, [41](#)
 - game_map.h, [46](#)
- displayPoint
 - point.c, [121](#)
 - point.h, [124](#)
- doubleCapacity
 - item_stack.c, [81](#)
 - item_stack.h, [85](#)
- dropOffLoc
 - item.h, [54](#)
- dropOffLocation
 - Item, [11](#)
- elem
 - boolean_matrix.h, [26](#)
 - location_matrix.h, [119](#)
- enqueue
 - item_queue.c, [74](#)
 - item_queue.h, [78](#)
- false
 - boolean.h, [23](#)
- Gadget, [8](#)
 - id, [8](#)
 - name, [9](#)
 - price, [9](#)
- gadget.c, [28](#)
 - isGadgetIdentical, [28](#)
 - KAIN_PEMBUNGKUS_WAKTU, [29](#)
 - MESIN_WAKTU, [29](#)
 - NULL_GADGET, [29](#)
 - PINTU_KEMANA_SAJA, [29](#)
 - SENER_PEMBESAR, [29](#)
 - SENER_PENGECIL, [30](#)
- gadget.h, [30](#)
 - id, [31](#)
 - isGadgetIdentical, [32](#)
 - KAIN_PEMBUNGKUS_WAKTU, [32](#)
 - MESIN_WAKTU, [32](#)
 - name, [31](#)
 - NULL_GADGET, [33](#)
 - PINTU_KEMANA_SAJA, [33](#)
 - price, [32](#)
 - SENER_PEMBESAR, [33](#)
 - SENER_PENGECIL, [33](#)
- gadget_list.c, [34](#)
 - displayGadget, [35](#)
 - getGadget, [35](#)
 - isGadgetListEmpty, [35](#)
 - isGadgetListFull, [36](#)
 - newGadgetList, [36](#)
 - setGadget, [36](#)
- gadget_list.h, [37](#)
 - displayGadget, [37](#)
 - getGadget, [38](#)
 - isGadgetListEmpty, [38](#)
 - isGadgetListFull, [39](#)
 - newGadgetList, [39](#)
 - setGadget, [39](#)
- GadgetList, [9](#)
 - contents, [9](#)
- game_map.c, [40](#)
 - _getAdjacentLocations, [41](#)
 - displayAdjacentLocation, [41](#)
 - displayGameMap, [41](#)
 - getLocationByCoord, [41](#)
 - getLocationById, [42](#)
 - getLocationBySymbol, [42](#)
 - isAdjacentTo, [43](#)
 - newGameMap, [43](#)
- game_map.h, [43](#)
 - adjMatrix, [44](#)
 - displayAdjacentLocation, [46](#)
 - displayGameMap, [46](#)
 - getLocationByCoord, [47](#)
 - getLocationById, [47](#)
 - getLocationBySymbol, [47](#)
 - isAdjacentTo, [48](#)
 - locList, [45](#)
 - locMatrix, [45](#)
 - mapLength, [46](#)
 - mapWidth, [46](#)
 - newGameMap, [48](#)
- GameMap, [10](#)
 - _adjacency, [10](#)
 - _locationMatrix, [10](#)
 - _locations, [10](#)
 - hSize, [11](#)
 - vSize, [11](#)
- gameMap
 - State, [21](#)
- getGadget
 - gadget_list.c, [35](#)
 - gadget_list.h, [38](#)
- getItem
 - item_list.c, [61](#)
 - item_list.h, [69](#)
- getLocationByCoord
 - game_map.c, [41](#)
 - game_map.h, [47](#)
- getLocationById
 - game_map.c, [42](#)
 - game_map.h, [47](#)
- getLocationBySymbol
 - game_map.c, [42](#)
 - game_map.h, [47](#)
- growList
 - location_list.c, [104](#)
 - location_list.h, [112](#)
- head
 - item_queue.h, [76](#)
- headIndex
 - item_queue.h, [77](#)
 - ItemQueue, [14](#)

HEAVY
 item.h, 54
hSize
 GameMap, 11
id
 Gadget, 8
 gadget.h, 31
 Location, 16
 location.h, 95
incrementCapacity
 item_stack.c, 81
 item_stack.h, 86
indexOfItem
 item_list.c, 61
 item_list.h, 69
inProgressList
 State, 21
insertItemAt
 item_list.c, 62
 item_list.h, 70
insertItemFirst
 item_list.c, 62
 item_list.h, 70
insertItemLast
 item_list.c, 62
 item_list.h, 70
insertLast
 location_list.c, 105
 location_list.h, 113
inventory
 State, 21
isAdjacentTo
 game_map.c, 43
 game_map.h, 48
isAt
 location.c, 89
 location.h, 96
isDropOffPlace
 Location, 16
isEmpty
 item_queue.c, 74
 item_queue.h, 78
isGadgetIdentical
 gadget.c, 28
 gadget.h, 32
isGadgetListEmpty
 gadget_list.c, 35
 gadget_list.h, 38
isGadgetListFull
 gadget_list.c, 36
 gadget_list.h, 39
isHeavyItem
 item.c, 50
 item.h, 56
isIndexEff
 location_list.c, 105
 location_list.h, 113
isIndexValid
 location_list.c, 105
 location_list.h, 113
isItemIdentical
 item.c, 50
 item.h, 56
isItemListEmpty
 item_list.c, 64
 item_list.h, 71
isItemListIndexValid
 item_list.c, 64
 item_list.h, 71
isLocationDefined
 location.c, 90
 location.h, 96
isLocationIdentical
 location.c, 90
 location.h, 97
isLocationListEmpty
 location_list.c, 106
 location_list.h, 114
isLocationListFull
 location_list.c, 106
 location_list.h, 114
isNormalItem
 item.c, 51
 item.h, 57
isPerishableItem
 item.c, 51
 item.h, 57
isPickUpPlace
 Location, 16
isPlayerPlace
 Location, 17
isPointBefore
 point.c, 121
 point.h, 124
isPointIdentical
 point.c, 122
 point.h, 125
isReachable
 Location, 17
isStackEmpty
 item_stack.c, 81
 item_stack.h, 86
isStackFull
 item_stack.c, 81
 item_stack.h, 86
isVIPItem
 item.c, 51
 item.h, 58
Item, 11
 dropOffLocation, 11
 orderTime, 12
 perishTime, 12
 pickUpLocation, 12
 type, 12
item.c, 49
 isHeavyItem, 50

- isItemIdentical, 50
- isNormalItem, 51
- isPerishableItem, 51
- isVIPItem, 51
- newItem, 52
- item.h, 52
 - dropOffLoc, 54
 - HEAVY, 54
 - isHeavyItem, 56
 - isItemIdentical, 56
 - isNormalItem, 57
 - isPerishableItem, 57
 - isVIPItem, 58
 - ItemType, 56
 - itemType, 54
 - newItem, 58
 - NORMAL, 54
 - orderTime, 54
 - PERISHABLE, 55
 - perishTime, 55
 - pickUpLoc, 55
 - UNTIMED, 55
 - VIP, 56
- item_list.c, 59
 - deleteItemAt, 60
 - deleteItemFirst, 60
 - deleteItemLast, 61
 - getItem, 61
 - indexOfItem, 61
 - insertItemAt, 62
 - insertItemFirst, 62
 - insertItemLast, 62
 - isItemListEmpty, 64
 - isItemListIndexValid, 64
 - itemListLength, 64
 - newItemList, 65
 - newItemListNode, 65
 - setItem, 65
- item_list.h, 66
 - deleteItemAt, 68
 - deleteItemFirst, 68
 - deleteItemLast, 68
 - getItem, 69
 - indexOfItem, 69
 - insertItemAt, 70
 - insertItemFirst, 70
 - insertItemLast, 70
 - isItemListEmpty, 71
 - isItemListIndexValid, 71
 - ItemList, 68
 - itemListLength, 71
 - newItemList, 72
 - newItemListNode, 72
 - next, 67
 - setItem, 72
 - value, 67
- item_queue.c, 73
 - dequeue, 74
 - enqueue, 74
 - isEmpty, 74
 - newItemQueue, 75
 - peekHeadTime, 75
- item_queue.h, 75
 - dequeue, 77
 - enqueue, 78
 - head, 76
 - headIndex, 77
 - isEmpty, 78
 - newItemQueue, 78
 - peekHeadTime, 79
 - tail, 77
 - tailIndex, 77
- item_stack.c, 80
 - _clampCapacity, 80
 - doubleCapacity, 81
 - incrementCapacity, 81
 - isStackEmpty, 81
 - isStackFull, 81
 - newItemStack, 82
 - pop, 82
 - push, 82
- item_stack.h, 83
 - _clampCapacity, 85
 - capacity, 84
 - doubleCapacity, 85
 - incrementCapacity, 86
 - isStackEmpty, 86
 - isStackFull, 86
 - ITEM_STACK_MAX_CAPACITY, 84
 - newItemStack, 86
 - pop, 87
 - push, 87
 - top, 84
 - topIndex, 85
- ITEM_STACK_MAX_CAPACITY
 - item_stack.h, 84
- ItemList
 - item_list.h, 68
- itemListLength
 - item_list.c, 64
 - item_list.h, 71
- ItemListNode, 12
 - next, 13
 - value, 13
- ItemQueue, 13
 - buffer, 14
 - headIndex, 14
 - tailIndex, 14
- ItemStack, 14
 - buffer, 15
 - capacity, 15
 - topIndex, 15
- ItemType
 - item.h, 56
- itemType
 - item.h, 54

KAIN_PEMBUNGKUS_WAKTU
 gadget.c, [29](#)
 gadget.h, [32](#)

IElem
 location_list.h, [110](#)

length
 location_list.c, [106](#)
 location_list.h, [114](#)

Location, [15](#)
 coordinate, [16](#)
 id, [16](#)
 isDropOffPlace, [16](#)
 isPickUpPlace, [16](#)
 isPlayerPlace, [17](#)
 isReachable, [17](#)
 symbol, [17](#)

location.c, [88](#)
 isAt, [89](#)
 isLocationDefined, [90](#)
 isLocationIdentical, [90](#)
 newLocation, [90](#)
 NULL_LOCATION, [93](#)
 setAsDropOffPlace, [91](#)
 setAsPickUpPlace, [91](#)
 setAsPlayerPlace, [91](#)
 setAsReachable, [92](#)
 toggleAsPlayerPlace, [92](#)
 unsetAsDropOffPlace, [92](#)
 unsetAsPickUpPlace, [92](#)
 unsetAsPlayerPlace, [93](#)
 unsetAsReachable, [93](#)
 writeLocationSymbol, [93](#)

location.h, [94](#)
 coord, [95](#)
 id, [95](#)
 isAt, [96](#)
 isLocationDefined, [96](#)
 isLocationIdentical, [97](#)
 newLocation, [97](#)
 NULL_LOCATION, [100](#)
 setAsDropOffPlace, [97](#)
 setAsPickUpPlace, [98](#)
 setAsPlayerPlace, [98](#)
 setAsReachable, [98](#)
 symbol, [96](#)
 toggleAsPlayerPlace, [99](#)
 unsetAsDropOffPlace, [99](#)
 unsetAsPickUpPlace, [99](#)
 unsetAsPlayerPlace, [99](#)
 unsetAsReachable, [100](#)
 writeLocationSymbol, [100](#)

location_list.c, [101](#)
 _getLocationByCoord, [102](#)
 _getLocationById, [103](#)
 _getLocationBySymbol, [103](#)
 compactList, [103](#)
 deallocateLocationList, [104](#)
 deleteLast, [104](#)

 growList, [104](#)
 insertLast, [105](#)
 isIndexEff, [105](#)
 isIndexValid, [105](#)
 isLocationListEmpty, [106](#)
 isLocationListFull, [106](#)
 length, [106](#)
 newLocationList, [107](#)
 shrinkList, [107](#)
 sortLocationListByCoord, [107](#)

location_list.h, [108](#)
 _getLocationByCoord, [110](#)
 _getLocationById, [111](#)
 _getLocationBySymbol, [111](#)
 buffer, [109](#)
 capacity, [109](#)
 compactList, [111](#)
 deallocateLocationList, [112](#)
 deleteLast, [112](#)
 growList, [112](#)
 insertLast, [113](#)
 isIndexEff, [113](#)
 isIndexValid, [113](#)
 isLocationListEmpty, [114](#)
 isLocationListFull, [114](#)
 IElem, [110](#)
 length, [114](#)
 neff, [110](#)
 newLocationList, [115](#)
 shrinkList, [115](#)
 sortLocationListByCoord, [115](#)

location_matrix.c, [116](#)
 ISetElem, [117](#)
 newLocationMatrix, [117](#)

location_matrix.h, [118](#)
 cols, [118](#)
 elem, [119](#)
 ISetElem, [119](#)
 newLocationMatrix, [120](#)
 rows, [119](#)

LocationList, [17](#)
 buffer, [18](#)
 capacity, [18](#)
 nEff, [18](#)

LocationMatrix, [18](#)
 colEff, [19](#)
 contents, [19](#)
 rowEff, [19](#)

locList
 game_map.h, [45](#)

locMatrix
 game_map.h, [45](#)

ISetElem
 location_matrix.c, [117](#)
 location_matrix.h, [119](#)

mapLength
 game_map.h, [46](#)

mapWidth

- game_map.h, 46
- MESIN_WAKTU
 - gadget.c, 29
 - gadget.h, 32
- moveItemToProgressList
 - state.h, 126
- name
 - Gadget, 9
 - gadget.h, 31
- nEff
 - LocationList, 18
- neff
 - location_list.h, 110
- newBooleanMatrix
 - boolean_matrix.c, 25
 - boolean_matrix.h, 27
- newGadgetList
 - gadget_list.c, 36
 - gadget_list.h, 39
- newGameMap
 - game_map.c, 43
 - game_map.h, 48
- newItem
 - item.c, 52
 - item.h, 58
- newItemList
 - item_list.c, 65
 - item_list.h, 72
- newItemListNode
 - item_list.c, 65
 - item_list.h, 72
- newItemQueue
 - item_queue.c, 75
 - item_queue.h, 78
- newItemStack
 - item_stack.c, 82
 - item_stack.h, 86
- newLocation
 - location.c, 90
 - location.h, 97
- newLocationList
 - location_list.c, 107
 - location_list.h, 115
- newLocationMatrix
 - location_matrix.c, 117
 - location_matrix.h, 120
- newPoint
 - point.c, 122
 - point.h, 125
- newState
 - state.h, 127
- next
 - item_list.h, 67
 - ItemListNode, 13
- NORMAL
 - item.h, 54
- NULL_GADGET
 - gadget.c, 29
- gadget.h, 33
- NULL_LOCATION
 - location.c, 93
 - location.h, 100
- ord
 - point.h, 124
- order
 - State, 22
- orderTime
 - Item, 12
 - item.h, 54
- peekHeadTime
 - item_queue.c, 75
 - item_queue.h, 79
- PERISHABLE
 - item.h, 55
- perishTime
 - Item, 12
 - item.h, 55
- pickUpLoc
 - item.h, 55
- pickUpLocation
 - Item, 12
- PINTU_KEMANA_SAJA
 - gadget.c, 29
 - gadget.h, 33
- Point, 19
 - x, 20
 - y, 20
- point.c, 120
 - displayPoint, 121
 - isPointBefore, 121
 - isPointIdentical, 122
 - newPoint, 122
- point.h, 123
 - abs, 123
 - displayPoint, 124
 - isPointBefore, 124
 - isPointIdentical, 125
 - newPoint, 125
 - ord, 124
- pop
 - item_stack.c, 82
 - item_stack.h, 87
- price
 - Gadget, 9
 - gadget.h, 32
- push
 - item_stack.c, 82
 - item_stack.h, 87
- reevaluate
 - state.h, 127
- rowEff
 - BooleanMatrix, 8
 - LocationMatrix, 19
- rows

- boolean_matrix.h, [26](#)
- location_matrix.h, [119](#)
- SENDER_PEMBESAR
 - gadget.c, [29](#)
 - gadget.h, [33](#)
- SENDER_PENGECIL
 - gadget.c, [30](#)
 - gadget.h, [33](#)
- setAsDropOffPlace
 - location.c, [91](#)
 - location.h, [97](#)
- setAsPickUpPlace
 - location.c, [91](#)
 - location.h, [98](#)
- setAsPlayerPlace
 - location.c, [91](#)
 - location.h, [98](#)
- setAsReachable
 - location.c, [92](#)
 - location.h, [98](#)
- setGadget
 - gadget_list.c, [36](#)
 - gadget_list.h, [39](#)
- setItem
 - item_list.c, [65](#)
 - item_list.h, [72](#)
- shrinkList
 - location_list.c, [107](#)
 - location_list.h, [115](#)
- sortLocationListByCoord
 - location_list.c, [107](#)
 - location_list.h, [115](#)
- State, [20](#)
 - bag, [21](#)
 - cash, [21](#)
 - currentLocation, [21](#)
 - gameMap, [21](#)
 - inProgressList, [21](#)
 - inventory, [21](#)
 - order, [22](#)
 - todoList, [22](#)
- state.h, [126](#)
 - moveItemToProgressList, [126](#)
 - newState, [127](#)
 - reevaluate, [127](#)
- symbol
 - Location, [17](#)
 - location.h, [96](#)
- tail
 - item_queue.h, [77](#)
- tailIndex
 - item_queue.h, [77](#)
 - ItemQueue, [14](#)
- todoList
 - State, [22](#)
- toggleAsPlayerPlace
 - location.c, [92](#)
- location.h, [99](#)
- top
 - item_stack.h, [84](#)
- topIndex
 - item_stack.h, [85](#)
 - ItemStack, [15](#)
- true
 - boolean.h, [24](#)
- type
 - Item, [12](#)
- unsetAsDropOffPlace
 - location.c, [92](#)
 - location.h, [99](#)
- unsetAsPickUpPlace
 - location.c, [92](#)
 - location.h, [99](#)
- unsetAsPlayerPlace
 - location.c, [93](#)
 - location.h, [99](#)
- unsetAsReachable
 - location.c, [93](#)
 - location.h, [100](#)
- UNTIMED
 - item.h, [55](#)
- value
 - item_list.h, [67](#)
 - ItemListNode, [13](#)
- VIP
 - item.h, [56](#)
- vSize
 - GameMap, [11](#)
- writeLocationSymbol
 - location.c, [93](#)
 - location.h, [100](#)
- x
 - Point, [20](#)
- y
 - Point, [20](#)