PG 369

- Klassendokumentation -

Fachbereich Informatik

Universität Dortmund

Inhaltsverzeichnis

1 PG369 Hierarchie-Verzeichnis

1.1 PG369 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

2 PG369 Compound-Verzeichnis

2.1 PG369 Übersicht

Hier folgt die Aufzählung aller Klassen, Strukturen und Varianten mit einer Kurzbeschreibung:

3 PG369 Datei-Verzeichnis

3.1 PG369 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

4 PG369 Klassen-Dokumentation

4.1 AusserhalbWertebereich Klassenreferenz

`#include <Fehler.h>`

4.1.1 Ausführliche Beschreibung

Wird erzeugt, wenn ein Parameter ausserhalb des zulaessigen Bereiches gesetzt wurde.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Fehler.h**

4.2 CAbschlussverknuepfung Klassenreferenz

`#include <CAbschlussverknuepfung.h>`

Öffentliche Datenelemente

- **CAbschlussverknuepfung** (**CNetzplan** * meinNetzplan)
- **CAbschlussverknuepfung** (**CNetzplan** * meinNetzplan, matrix *meinBedarf)
- **~CAbschlussverknuepfung** ()
- void **bildenMehrdeutigeLinienverknuepfung** ()
- void **durchfuehrenAufbruchsverknuepfung** ()
- void **bildenLinienendverknuepfung** ()
- void **loeschedoppelteBasislinien** ()

4.2.1 Ausführliche Beschreibung

Diese Klsse optimiert eine gegebene Liste von Linien (wobei es egal ist, ob diese Linien extern erstellt worden oder aber gerade in den vorhigen Phasen erzeugt worden sind). Zu diesem Zweck benutzt es die Funktionen Mehrfachverknuefung und Aufbruchverknuepfung um alle moeglichen Verknuepfungsarten zwischen den Linien untereinander zu bewerten, das Optimum auszuwaehlen und anschliessend die naechste, eine Verbesserung erzeugende Verknuepfung zu bestimmen.

4.2.2 Beschreibung der Konstruktoren und Destruktoren

4.2.2.1 CAbschlussverknuepfung::CAbschlussverknuepfung (CNetzplan * *meinNetzplan*)

Eingabe: **CNetzplan** (S. 121) als globale Datenstruktur

4.2.2.2 CAbschlussverknuepfung::CAbschlussverknuepfung (CNetzplan * *meinNetzplan*, matrix * *meinBedarf*)

4.2.2.3 CAbschlussverknuepfung::~CAbschlussverknuepfung ()

4.2.3 Dokumentation der Elementfunktionen

4.2.3.1 void CAbschlussverknuepfung::bildenLinienendverknuepfung ()

ist die Frage, ob wir diese Methode ueberhaupt implementieren, da sie kaum nutzen bringt... verschieben auf den 2. Prototypen

4.2.3.2 void CAbschlussverknuepfung::bildenMehrdeutigeLinienverknuepfung ()

Bei dieser Methode werden all diejenigen Kanten betrachtet, auf denen es mehrere verschiedene Verknuepfungsmoeglichkeiten zwischen den Linien gibt. Der Reihe nach werden all diese Verknuepfungen, die aktuell moeglich sind, mit Hilfe der Attraktivitiaetsberechnung bewertet, das Optimum ausgewaehlt und die entsprechende Kante fuer weitere Verknuepfungen gesperrt. Anschliessend wird der Algorithmus auf diesem Linienplan erneut aufgerufen und die naechste optimale Verknuepfung errechnet. Eine detailierte Beschreibung des verwendeten Algorithmus ist im Zwischenbericht auf Seite XYZ zu finden.

4.2.3.3 void CAbschlussverknuepfung::durchfuehrenAufbruchsverknuepfung ()

Bei dieser Methode erfolgt eine Kosten-Nutzen-Rechnung der Gestalt, dass der Attraktivitaets-Gewinn, der durch eine neue Verknuepfung entsteht, greoesser sein muss, als der Verlust, der dadurch entsteht dass eine andere Verknuepfung aufgebrochen wird, um so Gesamt-Laengen-Kapazitaet fuer die neue Verknuepfung zu erhalten. Als erstes wird deswegen entlang einer Linie (Grundlinie) eine sogenannte Verknuepfungsdiskussion durgefuehrt, um eine moegliche neue sinnvolle Verknuepfung mit einer anderen Linie zu finden. Ist diese bestimmt, so wird der Gewinn (durch die Attraktivitaetsberechnung) und die damit verbundenen Kosten (= neu benoetigte Entfernung fuer die Gesamtlinienlaenge) bestimmt. Dieses Verfahren setzt sich so lange fort, bis wieder eine optimale Verknuepfung diese Art gefunden worden ist. Anschliessend werden so lange diejenigen Kante bestimmt, die den guenstigsten Grad zwischen "Verlust" (=negativer Attraktivitaetsgewinn) und "Kostengewinn" (verfuegbare Entfernung fuer Gesamtlaenge) aufweisen, bis die benoetigte Kapazitaet erreicht ist, die die optimale Verknuepfung fordern wuerde. Anschliessend wird der Verlust durch diese Aufbrueche berechnet und wenn der Gewinn groesser als der Verlust ist, so wird diese Aufbruchverknuepfung durchgefuehrt. Eine detailierte Beschreibung des verwendeten Algorithmus ist im Zwischenbericht auf Seite XYZ zu finden.

4.2.3.4 void CAbschlussverknuepfung::loeschedoppelteBasislinien ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CAbschlussverknuepfung.h**
- **CAbschlussverknuepfung.cpp**

4.3 CAufbereitungsphase Klassenreferenz

```
#include <CAufbereitungsphase.h>
```

Öffentliche Datenelemente

- **CAufbereitungsphase** (**CNetzplan** * meinNetzplan, UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***> * wegenetz, matrix * qzmatrix)
- **~CAufbereitungsphase** ()
- void **starten** ()
- void **umlegen** ()
- void **reduzieren** ()

4.3.1 Ausführliche Beschreibung

Vorbereiten der Datenstrukturen auf die Verknuepfungsphase:

- **CNetzplan** (S. 121) aufbauen
- Verkehrsbedarfe als Verkehrsstrom an die Kanten legen
- Reduzieren von ueberfluessigen Knoten (nicht im ersten Prototyp)

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 CAufbereitungsphase::CAufbereitungsphase (CNetzplan * *meinNetzplan*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetz*, matrix * *qzmatrix*)

Entgegennahme der Parameter Eingaben:

- Wegenetzgraph
- Verkehrsbedarfsmatrix
- **CNetzplan** (S. 121) (mit leerem Graphen und leerer Liste)

4.3.2.2 CAufbereitungsphase::~CAufbereitungsphase ()

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 void CAufbereitungsphase::reduzieren ()

!!!!!!!!!!!! WIRD IM ERSTEN PROTOTYP NOCH NICHT VERWENDET!!!!!!!!!!!!

- Loeschen von Knoten, die keinen Umstieg ermoeglichen (= Knoten mit Grad 2)
- verhindern von Multigraphen: Wenn Knoten eigentlich geloescht werden soll, pruefen, ob schon eine Kante zwischen den beiden Nachbarkanten verlaeuft

4.3.3.2 void CAufbereitungsphase::starten (void)

Aufbau des Netzplangraphen Im grossen Kopie des Wegenetzgraphen mit folgenden Unterschieden:

- Entfernungen statt Fahrzeiten (teilen durch konstanten Faktor)
- an die Kanten zusaetzlich Verkehrsstrom und Anzahl der Basislinien

4.3.3.3 void CAufbereitungsphase::umlegen ()

Verkehrsstroeme an die Kanten legen:

- kuerzeste Wege zwischen allen Haltestellen ermitteln
- auf diese kuerzesten Wege den Verkehrsbedarf addieren

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CAufbereitungsphase.h**
- **CAufbereitungsphase.cpp**

4.4 CAufblinie Klassenreferenz

`#include <CAufblinie.h>`

Öffentliche Datenelemente

- **CAufblinie** ()
- **CAufblinie** (const CAufblinie&)
- CAufblinie& **operator=** (const CAufblinie&)
- int **holenID** ()
- bool **holenAnfang** ()
- list_item **holenLinieitem** ()
- int **holenaufbStrom** ()

- int **holenaufbLaenge** ()
- void **setzenID** (int id)
- void **setzenAnfang** (bool anf)
- void **setzenLinieitem** (list_item item)
- void **setzenhListe** (list<int>* einehListe)
- void **setzenaufbStrom** (int strom)
- void **setzenaufbLaenge** (int strom)
- list<int>* **holenhListe** ()
- **LEDA_MEMORY** (CAufblinie)

Freundbeziehungen

- istream& **operator>>** (istream&, CAufblinie&)
- ostream& **operator<<** (ostream&, const CAufblinie&)
- int **compare** (const CAufblinie* bLinie1, const CAufblinie* bLinie2)

4.4.1 Beschreibung der Konstruktoren und Destruktoren

4.4.1.1 CAufblinie::CAufblinie ()

Konstruktor

4.4.1.2 CAufblinie::CAufblinie (const CAufblinie & *originalLinie*)

Kopier Konstruktor

4.4.2 Dokumentation der Elementfunktionen

4.4.2.1 CAufblinie::LEDA_MEMORY (CAufblinie)

4.4.2.2 bool CAufblinie::holenAnfang ()

4.4.2.3 int CAufblinie::holenID ()

4.4.2.4 list_item CAufblinie::holenLinieitem ()

4.4.2.5 int CAufblinie::holenaufbLaenge ()

4.4.2.6 int CAufblinie::holenaufbStrom ()

4.4.2.7 list< int >* CAufblinie::holenhListe ()

4.4.2.8 CAufblinie & CAufblinie::operator= (const CAufblinie & *originalLinie*)

= Operator

4.4.2.9 void CAufblinie::setzenAnfang (bool *anf*)

4.4.2.10 void CAufblinie::setzenID (int *id*)

4.4.2.11 void CAufblinie::setzenLinieitem (list_item *item*)

4.4.2.12 void CAufblinie::setzenaufbLaenge (int *laenge*)

4.4.2.13 void CAufblinie::setzenaufbStrom (int *strom*)

4.4.2.14 void CAufblinie::setzenhListe (list< int >* *einehListe*)

4.4.3 Freundbeziehungen und Funktionsdokumentation

4.4.3.1 int compare (const CAufblinie * *bLinie1*, const CAufblinie * *bLinie2*) [friend]

4.4.3.2 ostream & operator<< (ostream & *os*, const CAufblinie & *AusgabeLinie*) [friend]

<< Operator

4.4.3.3 istream & operator>> (istream & *is*, CAufblinie & *ZielLinie*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CAufblinie.h**
- **CAufblinie.cpp**

4.5 CBasislinie Klassenreferenz

```
#include <CBasislinie.h>
```

Öffentliche Datenelemente

- **CBasislinie** ()
- **~CBasislinie** ()
- **CBasislinie** (const CBasislinie&)
- CBasislinie& **operator=** (const CBasislinie&)
- **CNetzplanKante*** **holenKante** (int nummer)
- void **setzenKante** (int nummer, **CNetzplanKante*** Kante)
- **CNetzplanKnoten*** **holenKnoten** (int nummer)
- void **setzenKnoten** (int nummer, **CNetzplanKnoten*** knoten)
- int **holenStrom** ()
- void **setzenStrom** (int strom)
- **LEDA_MEMORY** (CBasislinie)

Freundbeziehungen

- istream& **operator>>** (istream&, CBasislinie&)
- ostream& **operator<<** (ostream&, const CBasislinie&)
- int **compare** (const CBasislinie*, const CBasislinie*)

4.5.1 Ausführliche Beschreibung

Eine Basislinie ist ein Weg von genau zwei Kanten an einer Haltestelle im Wegenetz.

4.5.2 Beschreibung der Konstruktoren und Destruktoren

4.5.2.1 CBasislinie::CBasislinie ()

Default Konstruktor

4.5.2.2 CBasislinie::~CBasislinie ()

Default Destruktor

4.5.2.3 CBasislinie::CBasislinie (const CBasislinie & *originalLinie*)

Kopier Konstruktor

4.5.3 Dokumentation der Elementfunktionen

4.5.3.1 CBasislinie::LEDA_MEMORY (CBasislinie)

4.5.3.2 CNetzplanKante * CBasislinie::holenKante (int *nummer*)

gebe eine Kante zurueck

4.5.3.3 CNetzplanKnoten * CBasislinie::holenKnoten (int *nummer*)

gebe einen Knoten zurueck

4.5.3.4 int CBasislinie::holenStrom ()

gebe den Strom zurueck

4.5.3.5 CBasislinie & CBasislinie::operator= (const CBasislinie & *originalLinie*)

= Operator

4.5.3.6 void CBasislinie::setzenKante (int *nummer*, CNetzplanKante * *Kante*)

setze eine Kante

4.5.3.7 void CBasislinie::setzenKnoten (int *nummer*, CNetzplanKnoten * *knoten*)

setze einen Knoten

4.5.3.8 void CBasislinie::setzenStrom (int *strom*)

setze den Strom

4.5.4 Freundbeziehungen und Funktionsdokumentation

4.5.4.1 int compare (const CBasislinie * *bLinie1*, const CBasislinie * *bLinie2*) [friend]

Vergleichoperator um von gross nach klein zu sortieren

4.5.4.2 ostream & operator<< (ostream & *os*, const CBasislinie & *ausgabeLinie*) [friend]

<< Operator

4.5.4.3 istream & operator>> (istream & *is*, CBasislinie & *zielLinie*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CBasislinie.h**
- **CBasislinie.cpp**

4.6 CBevoelkerungsdichteElement Klassenreferenz

`#include <CBevoelkerungsdichteElement.h>`

Öffentliche Datenelemente

- **CPunkt* holenPunkt** ()
- double **holenDichte** ()
- double **holenRadius** ()
- void **setzenPunkt** (**CPunkt*** EinPunkt)
- void **setzenDichte** (double d)
- void **setzenRadius** (double r)
- **CBevoelkerungsdichteElement** (**CPunkt** *EinPunkt,double d, double r)
- **~CBevoelkerungsdichteElement** ()

4.6.1 Ausführliche Beschreibung

Ein Element der Bevoelkerungsdichtenliste. Hiermit soll fuer einen Matrix-Index der Bezug zum Koordinatensystem geschaffen werden.

4.6.2 Beschreibung der Konstruktoren und Destruktoren

4.6.2.1 CBevoelkerungsdichteElement::CBevoelkerungsdichte-Element (CPunkt * *EinPunkt*, double *d*, double *r*)

Konstruktor

4.6.2.2 CBevoelkerungsdichteElement::~CBevoelkerungsdichte-Element ()

Destruktor

4.6.3 Dokumentation der Elementfunktionen

4.6.3.1 double CBevoelkerungsdichteElement::holenDichte ()

Holen-Methode für die Dichte eines Elementes

Rückgabe:
Liefert die Anzahl der in dem angegebenen Raum lebenden Bevoelkerung.

4.6.3.2 CPunkt * CBevoelkerungsdichteElement::holenPunkt ()

"get"-Methode für den Punkt, der in einem Element gespeichert ist

Rückgabe:
Liefert den Mittelpunkt des Bevoelkerungsraumes.

4.6.3.3 double CBevoelkerungsdichteElement::holenRadius ()

Methode, um den Radius eines Elementes zu holen

Rückgabe:
Liefert den Umkreis, in welchem die angegebene Bevoelkerungsanzahl lebt.

4.6.3.4 void CBevoelkerungsdichteElement::setzenDichte (double *d*)

set-Methode, um die Dichte eines Elementes zu setzen

4.6.3.5 void CBevoelkerungsdichteElement::setzenPunkt (CPunkt * *EinPunkt*)

set-Methode für den Punkt eines BevölkerungsdichteElementes.

Parameter:
EinPunkt Der Mittelpunkt des zu betrachtenden Bevoelkerungsraumes.

4.6.3.6 void CBevoelkerungsdichteElement::setzenRadius (double *r*)

set-Methode, um den Radius eines Elementes zu setzen

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CBevoelkerungsdichteElement.h**
- **CBevoelkerungsdichteElement.cpp**

4.7 CBewertungsphase Klassenreferenz

`#include <CBewertungsphase.h>`

Öffentliche Datenelemente

- **CBewertungsphase** (**CNetzplan** * meinNetzplan, matrix * qzmatrix)
- **~CBewertungsphase** ()
- GRAPH<**CFPlanGraphKnoten***,**CFPlanGraphKante***>* **umlegenVerkehrsbedarfAufLinienplan** (**CGuetemasse*** guetemasse)

4.7.1 Ausführliche Beschreibung

Diese Klasse bewertet einen vorgegebenen Linienplan auf der Grundlage der Verkehrsbedarfs-Matrix, indem sie die Verkehrwuensche auf den vorgegebenen Linienplan umlegt und dabei relevante Werte mitprotokolliert. Ausserdem erzeugt sie den von der Fahrplangruppe verlangten Fahrplangenerator.

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 CBewertungsphase::CBewertungsphase (CNetzplan * *meinNetzplan*, matrix * *qzmatrix*)

Eingabe: CWegenetzplan (wenn es keine Reduktion gibt, dann entfaellt dieser Schritt), **CNetzplan** (S. 121), Verkehrsbedarfsmatrix

4.7.2.2 CBewertungsphase::~CBewertungsphase ()

4.7.3 Dokumentation der Elementfunktionen

4.7.3.1 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CBewertungsphase::umlegenVerkehrsbedarfAufLinienplan (CGuetemasse * *guetemasse*)

Alle Felder der Verkehrsbedarfsmatrix muessen "abgearbeitet" und auf den Linienplan umgelegt werden. Zu diesem Zweck wird zuerst die durch das Feld der Matrix (a,b) bestimmte Verkehrsbeziehung zwischen a und b auf dem kuerzesten Weg festgelegt. Diesmal wird aber im Gegensatz zu der Umlage in der Aufbereitungsphase nicht der gesamte Wegenetzplan, sondern nur diejenigen Kanten aus dem Wegenetzplan als Weg zugelassen, auf denen auch laut Linienplan tatsaechlich ein Bus faehrt. Wenn es entlang dieses kuerzesten Weges zu einem Umsteigevorgang zwischen zwei Buslinien kommt, dann wird zu der Fahrtzeit fuer diese Strecke eine Umsteige- und Wartezeit addiert, ansonsten nur die normale Fahrtzeit auf dieser Strecke. Ausserdem werden alle Fahrtwuensche entlang dieser Strecke zu den Fahrgaesten in den betroffenen Buslinien dazuaddiert. Was genau noch alles mit protokolliert wird (umsteige-anzahl) ist noch nicht festgelegt. Ausgabe: CFahrplangraph

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CBewertungsphase.h**
- **CBewertungsphase.cpp**

4.8 CBezeichner Klassenreferenz

`#include <CBezeichner.h>`

Öffentliche Datenelemente

- **setzeLiniennr1** ()
- **setzeLiniennr2** ()
- **setzeDauer** ()
- **holeLiniennr1** ()
- **holeLiniennr2** ()
- **holeDauer** ()

4.8.1 Dokumentation der Elementfunktionen

4.8.1.1 CBezeichner::holeDauer ()

4.8.1.2 CBezeichner::holeLiniennr1 ()

4.8.1.3 CBezeichner::holeLiniennr2 ()

4.8.1.4 CBezeichner::setzeDauer ()

4.8.1.5 CBezeichner::setzeLiniennr1 ()

4.8.1.6 CBezeichner::setzeLiniennr2 ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CBezeichner.h**

4.9 CBindungen Klassenreferenz

```
#include <CBindungen.h>
```

Öffentliche Datenelemente

- **setzeZusammenhangsKomponentenID** ()
- **setzeUmsteigeHaltestelle** ()
- **setzeLiniennr1** ()
- **setzeLiniennr2** ()
- **holeZusammenhangsKomponentenID** ()
- **holeUmsteigeHaltestelle** ()
- **holeLiniennr1** ()
- **holeLiniennr2** ()

4.9.1 Dokumentation der Elementfunktionen

4.9.1.1 CBindungen::holeLiniennr1 ()

4.9.1.2 CBindungen::holeLiniennr2 ()

4.9.1.3 CBindungen::holeUmsteigeHaltestelle ()

4.9.1.4 CBindungen::holeZusammenhangsKomponentenID ()

4.9.1.5 CBindungen::setzeLiniennr1 ()

4.9.1.6 CBindungen::setzeLiniennr2 ()

4.9.1.7 CBindungen::setzeUmsteigeHaltestelle ()

4.9.1.8 CBindungen::setzeZusammenhangsKomponentenID ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CBindungen.h**

4.10 CDatenbank Klassenreferenz

`#include <CDatenbank.h>`

Klassendiagramm für CDatenbank:

Öffentliche Datenelemente

- **CDatenbank** (char*, bool bbetrieb=false)
- virtual **~CDatenbank** ()
- virtual void **speichernGuetemasse** (**CGuetemasse** *)
- virtual **CGuetemasse*** **holenGuetemasse** ()
- virtual void **speichernDaten** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>* , matrix*)
- virtual void **speichernHaltestellenModulParameter** (int aufruf, int algorithmus, int zufallsgeneratorInitialisierung, double minRadius, double maxRadius, double verbindungsRadiusStadt, double verbindungsRadius-Land, int radiustyp, double haltestellenLage, double toleranzgrenze, int bevoelkerungsRaster, double koeffizientBevoelkerungsueberversorgung, double koeffizientBevoelkerungsunterversorgung, double koeffizient-Flaechenueberversorgung, double koeffizientFlaechenunterversorgung, double koeffizientFahrtwunscheinbeziehung, double koeffizient-Strassenanpassung, int AnzahlHaltestellen, bool AnzStrassenkarte, bool AnzBevoelkerungsversorgung, bool AnzFlaechenversorgung, bool Anzkomplettunterdruecken, int Strategie, int AnzahlEltern, int Anzahl-Kinder, int Generationen, bool AbbruchGueteunterschreitung, double GrenzwGueteunterschreitung)

- virtual void **holenHaltestellenModulParameter** (int aufruf, int& algorithmus, int& zufallsgeneratorInitialisierung, double& minRadius, double& maxRadius, double& verbindungsRadiusStadt, double& verbindungsRadiusLand, int& radiustyp, double& haltestellenLage, double& toleranzgrenze, int& bevoelkerungsRaster, double& koeffizientBevoelkerungsueberversorgung, double& koeffizientBevoelkerungsunterversorgung, double& koeffizientFlaechenueberversorgung, double& koeffizientFlaechenunterversorgung, double& koeffizientFahrtwunscheinbeziehung, double& koeffizientStrassenanpassung, int& AnzahlHaltestellen, bool& AnzStrassenkarte, bool& AnzBevoelkerungsversorgung, bool& AnzFlaechenversorgung, bool& Anzkomplettunterdruecken, int& Strategie, int& AnzahlEltern, int& AnzahlKinder, int& Generationen, bool& AbbruchGueteunterschreitung, double& GrenzwGueteunterschreitung)
- virtual void **holenDaten** (list<**CPunkt***>*&, list<list<**CPunkt***>*>*&, list<list<**CPunkt***>*>*& wege, list<**CBevoelkerungsdichteElement***>*&, matrix*&, list<**CQuelleZielListeElement***>*&, matrix*&, list<**CPendlerListeElement***>*&, list<list<**CPunkt***>*>*>*&)
- virtual void **speichernDaten** (GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>*, list<**CLinie***>*)
- virtual void **holenLinienplanModulParameter** (int aufruf, int& gesamtlaenge, int& maxLinienLaenge, int& maxLinienAnzahl, double& linienplanID, int& geschwindigkeit, int& algorithmus, int& anzeigegraphen, bool& EindeutigeLinienverkn, bool& MehrdeutigeLinienverkn, bool& Aufbruchsverknuepfung, bool& Linienendverknuepfung, bool& Basislinienwahl)
- virtual void **speichernLinienplanModulParameter** (int aufruf, int gesamtlaenge, int maxLinienLaenge, int maxLinienAnzahl, double linienplanID, int geschwindigkeit=15, int algorithmus=0, int anzeigegraphen=0, bool EindeutigeLinienverkn=false, bool MehrdeutigeLinienverkn=false, bool Aufbruchsverknuepfung=false, bool Linienendverknuepfung=false, bool Basislinienwahl=false)
- virtual void **holenDaten** (matrix*& , list<**CLinie***>*& , UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>*&)
- virtual matrix* **holenLinienplanQuelleZielMatrix** ()
- virtual UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>* **holenLinienplanWegenetzgraph** ()
- virtual void **speichernPflichtlinienDatenbank** (list<**CLinie***>* pflichtlinien)
- virtual void **speichernPflichtlinienGRASS** (list<**CLinie***>* pflichtlinien)
- virtual void **speichernEinePflichtlinie** (list<int>* pflichtlinie, int id)
- virtual list<**CLinie***>* **holenPflichtlinien** ()
- virtual void **speichernDaten** (array<list<three_tuple<int,int,int>* >* >*)
- virtual void **speichernDaten** (list<**CFahrplanElement***>*)
- virtual GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()

- virtual h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplanMindestumsteigezeit** ()
- virtual void **speichernFahrplanMindestumsteigezeit** (h_array<int,list<three_tuple<int,int,int>* >*>*)
- virtual h_array<int,int>* **fahrplanMindesthaltezeit** ()
- virtual void **speichernFahrplanMindesthaltezeiten** (h_array<int,int>*)
- virtual list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- virtual int **fahrplanMaximaleFahrzeuganzahl** ()
- virtual array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- virtual four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- virtual void **fahrplanerstellungWartezeiten** (double input_summeNetzbedingteWartezeiten, double input_wartezeitAufwand, double input_mindestWartezeitAufwand, double input_gesamtWarteaufwand)
- virtual void **fahrplanerstellungAnzahlBenoetigterFahrzeuge** (int input_fahrzeugAnzahl)
- virtual void **holenAusgangsloesung** (double& input_summeNetzbedingteWartezeiten, double& input_wartezeitAufwand, double& input_mindestWartezeitAufwand, double& input_gesamtWarteaufwand)
- virtual void **holenAnzahlFahrzeuge** (int& input_anzahlFahrzeuge)
- virtual list<**CFahrplanElement***>* **holenFahrplanHaltestellen** ()
- virtual list<**CFahrplanElement***>* **holenFahrplanLinien** ()
- virtual void **speichernFahrplanModulParameter** (int maxfahrzeuganzahl, int allgemeinertakt, int mindesthaltezeit, int mindestumsteigezeit)
- virtual void **holenFahrplanModulParameter** (int& maxfahrzeuganzahl, int& allgemeinertakt, int& mindesthaltezeit, int& mindestumsteigezeit)
- virtual void **speichernBevoelkerungsDichte** (list<**CBevoelkerungsdichteElement***>*)
- virtual void **speichernUrsprungsQZMatrix** (matrix* quelleZielMatrix, list<**CQuelleZielListeElement***>* quelleZielListe)
- virtual void **speichernPendlermatrix** (matrix* pendlerMatrix, list<**CPendlerListeElement***>* pendlerListe)
- virtual void **speichernVorgegebeneLinien** ()
- virtual int **holenKnotenID** (double, double)
- virtual **CPunkt*** **holenKnotenKoordinaten** (int)
- virtual list<int>* **holenLinienID** (int)
- virtual list<int>* **holenMoeglicheZielHaltestellen** (int)
- virtual int **holenZusammenhangsKomponentenID** (int,int)
- virtual list<int>* **holenEingehendeLinien** (int)
- virtual list<int>* **holenAusgehendeLinien** (int)
- virtual list<int>* **holenLinienHaltestellen** (int linienid)
- virtual void **speichernAusschnittStrassennetz** ()
- virtual void **loeschenLinie** (int id)
- virtual **CReisebewertung*** **routenplanung** (int starths, int endhs)
- virtual void **setzenAktuelleProjektID** (int id)

- virtual int **holenAktuelleProjektID** ()
- virtual void **setztenPhase** (int phase)
- virtual int **holenPhase** ()
- virtual int **anlegenNeuesProjekt** (int id, char* beschreibung, int phase)
- virtual void **holenProjektInformationen** (int id, char*& beschreibung, int& phase)
- virtual list<two_tuple<int,char*>*>* **holenAlleProjekt-Informationen** ()
- virtual int **holenAnzahlProjekte** ()
- virtual int **holenErsteFreieProjektID** ()
- virtual void **loeschenProjekt** (int id)
- virtual int **kopierenProjekt** (int quellid, int zielid, char* beschreibung=0)
- virtual int **anlegenNeuesProjektPG369** (int id, char* beschreibung, int phase)
- virtual void **holenProjektInformationenPG369** (int id, char*& beschreibung, int& phase)
- virtual list<two_tuple<int,char*>*>* **holenAlleProjekt-InformationenPG369** ()
- virtual int **holenErsteFreieProjektIDPG369** ()
- virtual void **loeschenProjektPG369** (int id)
- virtual int **kopierenProjektVonPG369** (int quellid, int zielid, char* beschreibung=0)
- virtual int **kopierenProjektNachPG369** (int quellid, int zielid, char* beschreibung=0)
- virtual void **speichernPflichthaltestellen** ()
- virtual void **holenPflichthaltestellen** ()
- virtual void **speichernAusschnitt** (double links, double oben, double rechts, double unten)
- virtual void **holenAusschnitt** (double& links, double& oben, double& rechts, double& unten)
- virtual void **aktuallisierenAnzeigen** (int phase)

Geschützte Datenelemente

- virtual bool **seinInRechteck** (double xKoord_grenzeLO, double yKoord_-grenzeLO, double xKoord_grenzeRU, double yKoord_grenzeRU, **CPunkt** * testPunkt)
- virtual list<list<**CPunkt*** >*>* **clippenStrassennetz** (list<list<**CPunkt*** > * > * strassennetz, double grenzeLinksOben_-X, double grenzeLinksOben_Y, double grenzeRechtsUnten_X, double grenzeRechtsUnten_Y)

Geschützte Attribute

- **CGRASSSchnittstelle* lnkCGRASSSchnittstelle**
- **CDBSchnittstelle* lnkCDBSchnittstelle**
- **CImport* lnkCImport**
- **CGuetemasse* temporaere_guetemasse**
- GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>* **temporaerer_wegenetzgraph**
- list<**CFahrplanElement***>* **derFahrplan**
- double **summeNetzbedingteWartezeiten**
- double **wartezeitAufwand**
- double **mindestWartezeitAufwand**
- double **gesamtWarteaufwand**
- int **anzahlFahrzeuge**
- bool **batchbetrieb**

4.10.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.10.2 Beschreibung der Konstruktoren und Destruktoren

4.10.2.1 CDatenbank::CDatenbank (char * *db_username*, bool *bbetrieb* = false)

4.10.2.2 CDatenbank::~CDatenbank () `[virtual]`

4.10.3 Dokumentation der Elementfunktionen

4.10.3.1 void CDatenbank::aktuallisierenAnzeigen (int *phase*) `[virtual]`

4.10.3.2 int CDatenbank::anlegenNeuesProjekt (int *id*, char * *beschreibung* = 0, int *phase* = 0) `[virtual]`

4.10.3.3 int CDatenbank::anlegenNeuesProjektPG369 (int *id*, char * *beschreibung*, int *phase*) `[virtual]`

4.10.3.4 list< list< CPunkt * >*>* CDatenbank::clippenStrassennetz (list< list< CPunkt * >* >* *strassennetz*, double *grenzeLinksOben_X*, double *grenzeLinksOben_Y*, double *grenzeRechtsUnten_X*, double *grenzeRechtsUnten_Y*) [protected, virtual]

Diese Methode clippt alle Strassen, die in dem durch die Grenzpunkte definierten Rechteck liegen

4.10.3.5 array< three_tuple< int,int,int >* >* CDatenbank::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Erneute Implementation in **CTestklasse0** (S. 180), **CTestklasse1** (S. 185), **CTestklasse2** (S. 188), **CTestklasse3** (S. 192), und **CTestklasse6** (S. 198).

4.10.3.6 list< four_tuple< int,int,int,int >* >* CDatenbank::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Erneute Implementation in **CTestklasse0** (S. 180), **CTestklasse1** (S. 185), **CTestklasse2** (S. 189), **CTestklasse3** (S. 192), und **CTestklasse6** (S. 198).

4.10.3.7 four_tuple< int,int,int,int >* CDatenbank::fahrplanIntervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Erneute Implementation in **CTestklasse0** (S. 181), **CTestklasse1** (S. 186), **CTestklasse2** (S. 189), **CTestklasse3** (S. 192), und **CTestklasse6** (S. 198).

4.10.3.8 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CDatenbank::fahrplanLinienplan () [virtual]

Erneute Implementation in **CTestklasse0** (S. 181), **CTestklasse1** (S. 186), **CTestklasse2** (S. 189), **CTestklasse3** (S. 192), und **CTestklasse6** (S. 199).

4.10.3.9 int CDatenbank::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Erneute Implementation in **CTestklasse0** (S. 181), **CTestklasse1** (S. 186), **CTestklasse2** (S. 189), **CTestklasse3** (S. 192), und **CTestklasse6** (S. 199).

4.10.3.10 h_array< int,int >* CDatenbank::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Erneute Implementation in **CTestklasse0** (S. 181), **CTestklasse1** (S. 186), **CTestklasse2** (S. 189), **CTestklasse3** (S. 193), und **CTestklasse6** (S. 199).

4.10.3.11 h_array< int,list< three_tuple< int,int,int >* >*>* CDatenbank::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Erneute Implementation in **CTestklasse0** (S. 182), **CTestklasse1** (S. 186), **CTestklasse2** (S. 190), **CTestklasse3** (S. 193), und **CTestklasse6** (S. 199).

4.10.3.12 void CDatenbank::fahrplanerstellungAnzahlBenoetigterFahrzeuge (int *input_fahrzeugAnzahl*) [virtual]

speichert die Anzahl benoetigter Fahrzeuge in der Datenbanktabelle Ausgangsloesung

4.10.3.13 void CDatenbank::fahrplanerstellungWartezeiten (double *input_summeNetzbedingteWartezeiten*, double *input_wartezeitAufwand*, double *input_mindestWartezeitAufwand*, double *input_gesamtWarteaufwand*) [virtual]

speichert die Wartezeiten in der Datenbanktabelle Ausgangsloesung

4.10.3.14 int CDatenbank::holenAktuelleProjektID () [virtual]

4.10.3.15 list< two_tuple< int,char *>*>* CDatenbank::holenAlleProjektInformationen () [virtual]

4.10.3.16 list< two_tuple< int,char *>*>* CDatenbank::holenAlleProjektInformationenPG369 () [virtual]

4.10.3.17 void CDatenbank::holenAnzahlFahrzeuge (int & *input_anzahlFahrzeuge*) [virtual]

diese Funktion holt die Anzahl der benoetigten Fahrzeuge:

4.10.3.18 int CDatenbank::holenAnzahlProjekte () [virtual]

4.10.3.19 void CDatenbank::holenAusgangsloesung (double & *input_summeNetzbedingteWartezeiten*, double & *input_wartezeitAufwand*, double & *input_mindestWartezeitAufwand*, double & *input_gesamtWarteaufwand*) [virtual]

diese Funktion holt die Ausgangsloesung (= Guetemasse des Fahrplans)

4.10.3.20 list< int >* CDatenbank::holenAusgehendeLinien (int *id*) [virtual]

4.10.3.21 void CDatenbank::holenAusschnitt (double & *links*, double & *oben*, double & *rechts*, double & *unten*) [virtual]

4.10.3.22 void CDatenbank::holenDaten (matrix *& *quelleZielMatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>*& *wegenetzGraph*) [virtual]

hole Daten fuer den Linienplan:

Parameter bedeuten dabei von links nach rechts: QuelleZielMatrix vorgegebene Linien Wegenetzgraph

Erneute Implementation in **CTestklasse0** (S. 182), **CTestklasse1** (S. 186), **CTestklasse2** (S. 190), und **CTestklasse3** (S. 193).

4.10.3.23 void CDatenbank::holenDaten (list< CPunkt *>*& *pflichthaltestellen*, list< list< CPunkt *>*>*& *strassennetz*, list< list< CPunkt *>*>*& *wege*, list< CBevoelkerungsdichteElement *>*& *bevoelkerungsdichte*, matrix *& *quellezielmatrix*, list< CQuelleZielListeElement *>*& *quellezielliste*, matrix *& *pendlermatrix*, list< CPendlerListeElement *>*& *pendlerliste*, list< list< CPunkt *>*>*& *gebaeude*) [virtual]

hole Daten fuer die Haltestellen:

Parameter bedeuten dabei von links nach rechts:

Parameter:

Pflichthaltestellen

Strassennetz

Bevoelkerungsdichte

Quelle-Ziel-Matrix

Quelle-Ziel-Liste

Pendlermatrix

Pendlerliste

Erneute Implementation in **CTestklasse0** (S. 182), **CTestklasse4** (S. 195), und **CTestklasse5** (S. 196).

4.10.3.24 list< int >* CDatenbank::holenEingehendeLinien (int *id*) [virtual]

4.10.3.25 int CDatenbank::holenErsteFreieProjektID () [virtual]

4.10.3.26 int CDatenbank::holenErsteFreieProjektIDPG369 () [virtual]

4.10.3.27 list< CFahrplanElement *>* CDatenbank::holenFahrplanHaltestellen () [virtual]

Diese Methode gibt eine Liste des Fahrplans aus Die Liste ist geordnet nach: HaltestellenID, LinienID, AbfahrtszeitStd, AbfahrtszeitMin

4.10.3.28 list< CFahrplanElement *>* CDatenbank::holenFahrplanLinien () [virtual]

Diese Methode gibt eine Liste des Fahrplans aus Die Liste ist geordnet nach: LinienID, HaltestellenID, AbfahrtszeitStd, AbfahrtszeitMin

4.10.3.29 void CDatenbank::holenFahrplanModulParameter (int & *maxfahrzeuganzahl*, int & *allgemeinertakt*, int & *mindesthaltezeit*, int & *mindestumsteigezeit*) [virtual]

4.10.3.30 CGuetemasse * CDatenbank::holenGuetemasse (void) [virtual]

Erneute Implementation in **CTestklasse1** (S. 187), **CTestklasse2** (S. 190), **CTestklasse3** (S. 193), und **CTestklasse6** (S. 199).

4.10.3.31 void CDatenbank::holenHaltestellenModulParameter (int *aufruf*, int & *algorithmus*, int & *zufallsgeneratorInitialisierung*, double & *minRadius*, double & *maxRadius*, double & *verbindungsRadiusStadt*, double & *verbindungsRadiusLand*, int & *radiustyp*, double & *haltestellenLage*, double & *toleranzgrenze*, int & *bevoelkerungsRaster*, double & *koeffizientBevoelkerungsueberversorgung*, double & *koeffizientBevoelkerungsunterversorgung*, double & *koeffizientFlaechenueberversorgung*, double & *koeffizientFlaechenunterversorgung*, double & *koeffizientFahrtwunscheinbeziehung*, double & *koeffizientStrassenanpassung*, int & *AnzahlHaltestellen*, bool & *AnzStrassenkarte*, bool & *AnzBevoelkerungsversorgung*, bool & *AnzFlaechenversorgung*, bool & *Anzkomplettunterdruecken*, int & *Strategie*, int & *AnzahlEltern*, int & *AnzahlKinder*, int & *Generationen*, bool & *AbbruchGueteunterschreitung*, double & *GrenzwGueteunterschreitung*) [virtual]

4.10.3.32 int CDatenbank::holenKnotenID (double *x*, double *y*) [virtual]

4.10.3.33 CPunkt * CDatenbank::holenKnotenKoordinaten (int *id*) [virtual]

4.10.3.34 list< int >* CDatenbank::holenLinienHaltestellen (int *linienid*) [virtual]

4.10.3.35 list< int >* CDatenbank::holenLinienID (int *id*) [virtual]

4.10.3.36 void CDatenbank::holenLinienplanModulParameter (int *aufruf*, int & *gesamtlaenge*, int & *maxLinienLaenge*, int & *maxLinienAnzahl*, double & *linienplanID*, int & *geschwindigkeit*, int & *algorithmus*, int & *anzeigegraphen*, bool & *EindeutigeLinienverkn*, bool & *MehrdeutigeLinienverkn*, bool & *Aufbruchsverknuepfung*, bool & *Linienendverknuepfung*, bool & *Basislinienwahl*) [virtual]

Erneute Implementation in **CTestklasse0** (S. 183).

4.10.3.37 matrix * CDatenbank::holenLinienplanQuelleZielMatrix () [virtual]

Erneute Implementation in **CTestklasse0** (S. 183), **CTestklasse1** (S. 187), **CTestklasse2** (S. 190), und **CTestklasse3** (S. 193).

4.10.3.38 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CDatenbank::holenLinienplanWegenetzgraph () [virtual]

Erneute Implementation in **CTestklasse0** (S. 184), **CTestklasse1** (S. 187), **CTestklasse2** (S. 190), und **CTestklasse3** (S. 193).

4.10.3.39 list< int >* CDatenbank::holenMoeglicheZielHaltestellen (int *id*) [virtual]

4.10.3.40 void CDatenbank::holenPflichthaltestellen (void) [virtual]

4.10.3.41 list< CLinie *>* CDatenbank::holenPflichtlinien () [virtual]

4.10.3.42 int CDatenbank::holenPhase () [virtual]

4.10.3.43 void CDatenbank::holenProjektInformationen (int *id*, char *& *beschreibung*, int & *phase*) [virtual]

4.10.3.44 void CDatenbank::holenProjektInformationenPG369 (int *id*, char *& *beschreibung*, int & *phase*) [virtual]

4.10.3.45 int CDatenbank::holenZusammenhangsKomponentenID (int *h*, int *l1*) [virtual]

4.10.3.46 int CDatenbank::kopierenProjekt (int *quellid*, int *zielid*, char * *beschreibung* = 0) [virtual]

4.10.3.47 int CDatenbank::kopierenProjektNachPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0) [virtual]

4.10.3.48 int CDatenbank::kopierenProjektVonPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0) [virtual]

4.10.3.49 void CDatenbank::loeschenLinie (int *id*) [virtual]

Das Strassennetz clippen!

4.10.3.50 void CDatenbank::loeschenProjekt (int *id*) [virtual]

4.10.3.51 void CDatenbank::loeschenProjektPG369 (int *id*) [virtual]

4.10.3.52 CReisebewertung * CDatenbank::routenplanung (int *starths*, int *endhs*) [virtual]

4.10.3.53 bool CDatenbank::seinInRechteck (double *xKoord_grenzeLO*, double *yKoord_grenzeLO*, double *xKoord_grenzeRU*, double *yKoord_grenzeRU*, CPunkt * *testPunkt*) [protected, virtual]

Testet einen Punkt, ob dieser in einem durch die Grenzpunkte definierten Rechteck liegt

4.10.3.54 void CDatenbank::setzenAktuelleProjektID (int *id*) [virtual]

4.10.3.55 void CDatenbank::setztenPhase (int *phase*) [virtual]

4.10.3.56 void CDatenbank::speichernAusschnitt (double *links*, double *oben*, double *rechts*, double *unten*) [virtual]

4.10.3.57 void CDatenbank::speichernAusschnittStrassennetz () [virtual]

Erneute Implementation in **CTestklasse0** (S. 184), **CTestklasse4** (S. 195), und **CTestklasse5** (S. 197).

4.10.3.58 void CDatenbank::speichernBevoelkerungsDichte (list< CBevoelkerungsdichteElement *>* *bd*) [virtual]

Folgende Funktionen dienen der GUI: sie kann auf diese Weise Daten in die Datenbank schieben und auch wieder herausholen

4.10.3.59 void CDatenbank::speichernDaten (list< CFahrplanElement *>* *fahrplan* = 0) [virtual]

4.10.3.60 void CDatenbank::speichernDaten (array< list< three_tuple< int,int,int >* >* >* *zusammenhangskomponenten* = 0) [virtual]

4.10.3.61 void CDatenbank::speichernDaten (GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *fpgraph* = 0, list< CLinie *>* *buslinien* = 0) [virtual]

4.10.3.62 void CDatenbank::speichernDaten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wgraph*, matrix * *qz-matrix*) [virtual]

4.10.3.63 void CDatenbank::speichernEinePflichtlinie (list< int >* *pflichtlinie* = 0, int *id* = 0) [virtual]

4.10.3.64 void CDatenbank::speichernFahrplanMindesthaltezeiten (h_array< int,int >* *mh*) [virtual]

speichert die Mindestumsteigezeit

4.10.3.65 void CDatenbank::speichernFahrplanMindestumsteigezeit (h_array< int,list< three_tuple< int,int,int >* >*>* *mu*) [virtual]

speichert die Mindestumsteigezeit

4.10.3.66 void CDatenbank::speichernFahrplanModulParameter (int *maxfahrzeuganzahl*, int *allgemeinertakt*, int *mindesthaltezeit*, int *mindestumsteigezeit*) [virtual]

4.10.3.67 void CDatenbank::speichernGuetemasse (CGuetemasse * *guetemasse*) [virtual]

Diese Methode schneidet alle Strassen, die in dem durch die Grenzpunkte definierten Rechteck liegen, ab.

4.10.3.68 void CDatenbank::speichernHaltestellenModulParameter (int *aufruf*, int *algorithmus*, int *zufallsgeneratorInitialisierung*, double *minRadius*, double *maxRadius*, double *verbindungsRadiusStadt*, double *verbindungsRadiusLand*, int *radiustyp*, double *haltestellenLage*, double *toleranzgrenze*, int *bevoelkerungsRaster*, double *koeffizientBevoelkerungsueberversorgung*, double *koeffizientBevoelkerungsunterversorgung*, double *koeffizientFlaechenueberversorgung*, double *koeffizientFlaechenunterversorgung*, double *koeffizientFahrtwunscheinbeziehung*, double *koeffizientStrassenanpassung*, int *AnzahlHaltestellen*, bool *AnzStrassenkarte*, bool *AnzBevoelkerungsversorgung*, bool *AnzFlaechenversorgung*, bool *Anzkomplettunterdruecken*, int *Strategie*, int *AnzahlEltern*, int *AnzahlKinder*, int *Generationen*, bool *AbbruchGueteunterschreitung*, double *GrenzwGueteunterschreitung*) [virtual]

4.10.3.69 void CDatenbank::speichernLinienplanModulParameter (int *aufruf*, int *gesamtlaenge*, int *maxLinienLaenge*, int *maxLinienAnzahl*, double *linienplanID*, int *geschwindigkeit* = 15, int *algorithmus* = 0, int *anzeigegraphen* = 0, bool *EindeutigeLinienverkn* = false, bool *MehrdeutigeLinienverkn* = false, bool *Aufbruchsverknuepfung* = false, bool *Linienendverknuepfung* = false, bool *Basislinienwahl* = false) [virtual]

4.10.3.70 void CDatenbank::speichernPendlermatrix (matrix * *pendlerMatrix*, list< CPendlerListeElement *>* *pendlerListe*) [virtual]

4.10.3.71 void CDatenbank::speichernPflichthaltestellen () [virtual]

4.10.3.72 void CDatenbank::speichernPflichtlinienDatenbank (list< CLinie *>* *pflichtlinien*) [virtual]

4.10.3.73 void CDatenbank::speichernPflichtlinienGRASS (list< CLinie *>* *pflichtlinien*) [virtual]

4.10.3.74 void CDatenbank::speichernUrsprungsQZMatrix (matrix * *quelleZielMatrix*, list< CQuelleZielListeElement *>* *quelleZielListe*) [virtual]

4.10.3.75 void CDatenbank::speichernVorgegebeneLinien () [virtual]

4.10.4 Dokumentation der Datenelemente

4.10.4.1 int CDatenbank::anzahlFahrzeuge [protected]

4.10.4.2 bool CDatenbank::batchbetrieb [protected]

4.10.4.3 list< CFahrplanElement *>* CDatenbank::derFahrplan [protected]

4.10.4.4 double CDatenbank::gesamtWarteaufwand [protected]

4.10.4.5 CDBSchnittstelle * CDatenbank::lnkCDBSchnittstelle [protected]

4.10.4.6 CGRASSSchnittstelle * CDatenbank::lnkCGRASSSchnittstelle [protected]

4.10.4.7 CImport * CDatenbank::lnkCImport [protected]

4.10.4.8 double CDatenbank::mindestWartezeitAufwand [protected]

4.10.4.9 double CDatenbank::summeNetzbedingteWartezeiten [protected]

4.10.4.10 CGuetemasse * CDatenbank::temporaere_guetemasse [protected]

4.10.4.11 GRAPH< CWegenetzKnoten *,CWegenetzKante *>* CDatenbank::temporaerer_wegenetzgraph [protected]

4.10.4.12 double CDatenbank::wartezeitAufwand [protected]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CDatenbank.h**
- **CDatenbank.cpp**

4.11 CDBSchnittstelle Klassenreferenz

`#include <CDBSchnittstelle.h>`

Öffentliche Datenelemente

- **CDBSchnittstelle** (char*, bool bbetrieb=false)
- **~CDBSchnittstelle** ()
- int **holenKnotenID** (double, double)
- **CPunkt* holenKnotenKoordinaten** (int)
- list<int>* **holenLinienID** (int)
- list<int>* **holenMoeglicheZielHaltestellen** (int)
- int **holenZusammenhangsKomponentenID** (int,int)
- list<int>* **holenEingehendeLinien** (int)
- list<int>* **holenAusgehendeLinien** (int)
- list<int>* **holenLinienHaltestellen** (int linienid)
- void **setzenAktuelleProjektID** (int id)
- int **holenAktuelleProjektID** ()
- void **setztenPhase** (int phase)
- int **holenPhase** ()
- int **anlegenNeuesProjekt** (int id, char* beschreibung, int phase)
- void **holenProjektInformationen** (int id, char*& beschreibung, int& phase)
- list<two_tuple<int,char*>*>* **holenAlleProjektInformationen** ()
- int **holenAnzahlProjekte** ()
- int **holenErsteFreieProjektID** ()
- void **loeschenProjekt** (int id)
- int **kopierenProjekt** (int quellid, int zielid, char* beschreibung=0)
- int **anlegenNeuesProjektPG369** (int id, char* beschreibung, int phase)

- void **holenProjektInformationenPG369** (int id, char*& beschreibung, int& phase)
- list<two_tuple<int,char*>*>* **holenAlleProjektInformationenPG369** ()
- int **holenErsteFreieProjektIDPG369** ()
- void **loeschenProjektPG369** (int id)
- int **kopierenProjektVonPG369** (int quellid, int zielid, char* beschreibung=0)
- int **kopierenProjektNachPG369** (int quellid, int zielid, char* beschreibung=0)
- void **speichernPflichthaltestellen** (list<**CPunkt***>* pflichthaltestellen)
- list<**CPunkt***>* **holenPflichthaltestellen** ()
- void **speichernAusschnitt** (double links, double oben, double rechts, double unten)
- void **holenAusschnitt** (double& links, double& oben, double& rechts, double& unten)
- matrix* **holenQuelleZielMatrix** ()
- void **speichernQuelleZielMatrix** (matrix*)
- matrix* **holenUrsprungsQZMatrix** ()
- void **speichernUrsprungsQZMatrix** (matrix*)
- list<**CQuelleZielListeElement***>* **holenQuelleZielListe** ()
- void **speichernQuelleZielListe** (list<**CQuelleZielListeElement***>*)
- list<**CQuelleZielListeElement***>* **holenUrsprungsQZListe** ()
- void **speichernUrsprungsQZListe** (list<**CQuelleZielListeElement***>*)
- list<**CWegenetzKnoten***>* **holenWegenetzKnoten** ()
- void **speichernWegenetzKnoten** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*)
- void **holenWegenetzKanten** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>* wegenetzgraph=0)
- void **speichernWegenetzKanten** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*)
- list<**CPunkt***>* **holenKantenStrasse** (int kantenid)
- void **speichernKantenStrasse** (list<**CPunkt***>*, int)
- list<**CLinie***>* **holenBusLinien** (int plannummer=0)
- void **speichernBusLinien** (GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* fpgraph=0,list<**CLinie***>* buslinien=0, int planid=0)
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **holenFahrplanGraph** (int planid=0)
- void **speichernGuetemasse** (**CGuetemasse** *)
- **CGuetemasse*** **holenGuetemasse** ()
- void **holenHaltestellenModulParameter** (int aufruf, int& algorithmus, int& zufallsgeneratorInitialisierung, double& minRadius, double& maxRadius, double& verbindungsRadiusStadt, double& verbindungsRadiusLand, int& radiustyp, double&

haltestellenLage, double& toleranzgrenze, int& bevoelkerungs-Raster, double& koeffizientBevoelkerungsueberversorgung, double& koeffizientBevoelkerungsunterversorgung, double& koeffizient-Flaechenueberversorgung, double& koeffizientFlaechenunterversorgung, double& koeffizientFahrtwunscheinbeziehung, double& koeffizient-Strassenanpassung, int& AnzahlHaltestellen, bool& AnzStrassenkarte, bool& AnzBevoelkerungsversorgung, bool& AnzFlaechenversorgung, bool& Anzkomplettunterdruecken, int& Strategie, int& Anzahl-Eltern, int& AnzahlKinder, int& Generationen, bool& Abbruch-Gueteunterschreitung, double& GrenzwGueteunterschreitung)

- void **speichernHaltestellenModulParameter** (int aufruf, int algorithmus, int zufallsgeneratorInitialisierung, double minRadius, double maxRadius, double verbindungsRadiusStadt, double verbindungsRadius-Land, int radiustyp, double haltestellenLage, double toleranzgrenze, int bevoelkerungsRaster, double koeffizientBevoelkerungsueberversorgung, double koeffizientBevoelkerungsunterversorgung, double koeffizient-Flaechenueberversorgung, double koeffizientFlaechenunterversorgung, double koeffizientFahrtwunscheinbeziehung, double koeffizient-Strassenanpassung, int AnzahlHaltestellen, bool AnzStrassenkarte, bool AnzBevoelkerungsversorgung, bool AnzFlaechenversorgung, bool Anzkomplettunterdruecken, int Strategie, int AnzahlEltern, int Anzahl-Kinder, int Generationen, bool AbbruchGueteunterschreitung, double GrenzwGueteunterschreitung)
- list<**CBevoelkerungsdichteElement**∗>∗ **holenBevoelkerungs-Dichte** ()
- void **speichernBevoelkerungsDichte** (list<**CBevoelkerungsdichte-Element**∗>∗)
- matrix∗ **holenPendlermatrix** ()
- void **speichernPendlermatrix** (matrix∗)
- list<**CPendlerListeElement**∗>∗ **holenPendlerliste** ()
- void **speichernPendlerliste** (list<**CPendlerListeElement**∗>∗)
- void **holenLinienplanModulParameter** (int aufruf, int& gesamtlaenge, int& maxLinienLaenge, int& maxLinienAnzahl, double& linienplan-ID, int& geschwindigkeit, int& algorithmus, int& anzeigegraphen, bool& EindeutigeLinienverkn, bool& MehrdeutigeLinienverkn, bool& Aufbruchsverknuepfung, bool& Linienendverknuepfung, bool& Basislinienwahl)
- void **speichernLinienplanModulParameter** (int aufruf, int gesamtlaenge, int maxLinienLaenge, int maxLinienAnzahl, double linienplanID, int geschwindigkeit=15, int algorithmus=0, int anzeigegraphen=0, bool EindeutigeLinienverkn=false, bool MehrdeutigeLinienverkn=false, bool Aufbruchsverknuepfung=false, bool Linienendverknuepfung=false, bool Basislinienwahl=false)
- **CLinie**∗ **holenLinienweg** (int id)
- void **speichernLinienweg** (**CLinie**∗)
- list<**CLinie**∗>∗ **holenLinienplan** ()
- void **speichernLinienplan** (list<**CLinie**∗>∗)
- list<**CLinie**∗>∗ **holenPflichtlinien** ()
- **CLinie**∗ **holenPflichtlinie** (int linienid)

- void **speichernPflichtlinien** (list<**CLinie**∗>∗ pflichtlinien)
- h_array<int,two_tuple<int,int>∗ >∗ **holenKnotenWerte** ()
- void **speichernKnotenWerte** (h_array<int,two_tuple<int,int>∗ >∗)
- void **loeschenKnotenWerte** ()
- h_array<int,three_tuple<int,int,bool>∗ >∗ **holenLinienwerte** ()
- void **speichernLinienwerte** (h_array<int,three_tuple<int,int,bool>∗ >∗)
- void **loeschenLinienwerte** ()
- list<three_tuple<int,int,int>∗ >∗ **holenAnschluss** ()
- void **speichernAnschluss** (list<three_tuple<int,int,int>∗ >∗)
- void **loeschenAnschluss** ()
- list<**CFahrplanElement**∗>∗ **holenFahrplanHaltestellen** ()
- list<**CFahrplanElement**∗>∗ **holenFahrplanLinien** ()
- void **loeschenFahrplan** ()
- void **speichernFahrplan** (list<**CFahrplanElement**∗>∗)
- array<list<three_tuple<int,int,int>∗ >∗ >∗ **holenZusammenhangskomponenten** ()
- void **speichernZusammenhangskomponenten** (array<list<three_tuple<int,int,int>∗ >∗ >∗)
- void **loeschenZusammenhangskomponenten** ()
- void **holenAusgangsloesung** (int,double&,double&,double&,int&)
- void **speichernAusgangsloesung** (int,double,double,double,int)
- void **loeschenAusgangsloesung** (int)
- list<three_tuple<int,int,int>∗ >∗ **holenAusgangsloesungHaltestelle** (int)
- void **speichernAusgangsloesungHaltestelle** (int,list<three_tuple<int,int,int>∗>∗)
- void **loeschenAusgangsloesungHaltestelle** (int)
- void **holenLinienplanBewertung** (int,int&,int&)
- void **speichernLinienplanBewertung** (int,int,int)
- void **loeschenLinienplanBewertung** ()
- void **speichernMindestumsteigezeit** (h_array<int,list<three_tuple<int,int,int>∗ >∗>∗)
- h_array<int,list<three_tuple<int,int,int>∗ >∗>∗ **holenMindestumsteigezeit** ()
- void **speichernMindesthaltezeiten** (h_array<int,int>∗)
- h_array<int,int>∗ **holenMindesthaltezeiten** ()
- void **fahrplanerstellungWartezeiten** (double summeNetzbedingteWartezeiten, double wartezeitAufwand, double mindestWartezeitAufwand, double gesamtWarteaufwand)
- void **fahrplanerstellungAnzahlBenoetigterFahrzeuge** (int fahrzeugAnzahl)
- void **speichernFahrplanModulParameter** (int maxfahrzeuganzahl, int allgemeinertakt, int mindesthaltezeit, int mindestumsteigezeit)
- void **holenFahrplanModulParameter** (int& maxfahrzeuganzahl, int& allgemeinertakt, int& mindesthaltezeit, int& mindestumsteigezeit)

4.11.1 Ausführliche Beschreibung

stellt die Schnittstelle zur Oracle-Datenbank her

4.11.2 Beschreibung der Konstruktoren und Destruktoren

4.11.2.1 CDBSchnittstelle::CDBSchnittstelle (char * *db_username*, bool *bbetrieb* = false)

4.11.2.2 CDBSchnittstelle::~CDBSchnittstelle ()

4.11.3 Dokumentation der Elementfunktionen

4.11.3.1 int CDBSchnittstelle::anlegenNeuesProjekt (int *id*, char * *beschreibung* = NULL, int *phase* = 0)

4.11.3.2 int CDBSchnittstelle::anlegenNeuesProjektPG369 (int *id*, char * *beschreibung* = NULL, int *phase* = 0)

4.11.3.3 void CDBSchnittstelle::fahrplanerstellungAnzahlBenoetigterFahrzeuge (int *fahrzeugAnzahl*)

speichert die Anzahl benoetigter Fahrzeuge in der Datenbanktabelle Ausgangsloesung

4.11.3.4 void CDBSchnittstelle::fahrplanerstellungWartezeiten (double *summeNetzbedingteWartezeiten*, double *wartezeitAufwand*, double *mindestWartezeitAufwand*, double *gesamtWarteaufwand*)

speichert die Wartezeiten in der Datenbanktabelle Ausgangsloesung

4.11.3.5 int CDBSchnittstelle::holenAktuelleProjektID ()

4.11.3.6 list< two_tuple< int,char *>*>* CDBSchnittstelle::holenAlleProjektInformationen ()

4.11.3.7 list< two_tuple< int,char *>*>* CDBSchnittstelle::holenAlleProjektInformationenPG369 ()

4.11.3.8 list< three_tuple< int,int,int >* >* CDBSchnittstelle::holenAnschluss ()

Diese Methode holt alle direkten Anschluesse aus der Datenbank

4.11.3.9 int CDBSchnittstelle::holenAnzahlProjekte ()

4.11.3.10 void CDBSchnittstelle::holenAusgangsloesung (int *loesungsID*, double & *wartezeitaufwand*, double & *mindestwartezeit*, double & *gesamtwartezeit*, int & *anzahl_busse*)

Diese Methode holt die Werte einer Ausgangsloesung aus der Tabelle Ausgangsloesung die Werte sind LoesungsID, Wartezeitaufwand in Min, Mindestwartezeitaufwand in Min und Gesamtwartezeitaufwand in Min

4.11.3.11 list< three_tuple< int,int,int >* >* CDBSchnittstelle::holenAusgangsloesungHaltestelle (int *loesungsID*)

Diese Methode holt aus der Tabelle AusgangsloesungHaltestelle die Liste der Haltestellenwerte die zur uebergebenen LoesungsID gehoeren

4.11.3.12 list< int >* CDBSchnittstelle::holenAusgehendeLinien (int *hs_input*)

4.11.3.13 void CDBSchnittstelle::holenAusschnitt (double & *links*, double & *oben*, double & *rechts*, double & *unten*)

4.11.3.14 list< CBevoelkerungsdichteElement *>* CDBSchnittstelle::holenBevoelkerungsDichte ()

4.11.3.15 list< CLinie *>* CDBSchnittstelle::holenBusLinien (int *plannummer* = 0)

4.11.3.16 list< int >* CDBSchnittstelle::holenEingehendeLinien (int *hs_input*)

4.11.3.17 int CDBSchnittstelle::holenErsteFreieProjektID ()

4.11.3.18 int CDBSchnittstelle::holenErsteFreieProjektIDPG369 ()

4.11.3.19 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CDBSchnittstelle::holenFahrplanGraph (int *planid* = 0)

4.11.3.20 list< CFahrplanElement *>* CDBSchnittstelle::holenFahrplanHaltestellen ()

Diese Methode gibt eine Liste des Fahrplans aus Die Liste ist geordnet nach: HaltestellenID, LinienID, AbfahrtszeitStd, AbfahrtszeitMin

4.11.3.21 list< CFahrplanElement *>* CDBSchnittstelle::holenFahrplanLinien ()

Diese Methode gibt eine Liste des Fahrplans aus Die Liste ist geordnet nach: LinienID, HaltestellenID, AbfahrtszeitStd, AbfahrtszeitMin

4.11.3.22 void CDBSchnittstelle::holenFahrplanModulParameter (int & *maxfahrzeuganzahl*, int & *allgemeinertakt*, int & *mindesthaltezeit*, int & *mindestumsteigezeit*)

4.11.3.23 CGuetemasse * CDBSchnittstelle::holenGuetemasse (void)

4.11.3.24 void CDBSchnittstelle::holenHaltestellenModulParameter (int *aufruf*, int & *algorithmus*, int & *zufallsgeneratorInitialisierung*, double & *minRadius*, double & *maxRadius*, double & *verbindungsRadiusStadt*, double & *verbindungsRadiusLand*, int & *radiustyp*, double & *haltestellenLage*, double & *toleranzgrenze*, int & *bevoelkerungsRaster*, double & *koeffizientBevoelkerungsueberversorgung*, double & *koeffizientBevoelkerungsunterversorgung*, double & *koeffizientFlaechenueberversorgung*, double & *koeffizientFlaechenunterversorgung*, double & *koeffizientFahrtwunscheinbeziehung*, double & *koeffizientStrassenanpassung*, int & *AnzahlHaltestellen*, bool & *AnzStrassenkarte*, bool & *AnzBevoelkerungsversorgung*, bool & *AnzFlaechenversorgung*, bool & *Anzkomplettunterdruecken*, int & *Strategie*, int & *AnzahlEltern*, int & *AnzahlKinder*, int & *Generationen*, bool & *AbbruchGueteunterschreitung*, double & *GrenzwGueteunterschreitung*)

4.11.3.25 list< CPunkt *>* CDBSchnittstelle::holenKantenStrasse (int *kantenid* = 0)

4.11.3.26 int CDBSchnittstelle::holenKnotenID (double *x_eingabe*, double *y_eingabe*)

4.11.3.27 CPunkt * CDBSchnittstelle::holenKnotenKoordinaten (int *id_eingabe*)

4.11.3.28 h_array< int,two_tuple< int,int >* >* CDBSchnittstelle::holenKnotenWerte ()

Diese Methode holt alle Knotenwerte aus der Datenbank

4.11.3.29 list< int >* CDBSchnittstelle::holenLinienHaltestellen (int *linienid*)

4.11.3.30 list< int >* CDBSchnittstelle::holenLinienID (int *hs_input*)

4.11.3.31 list< CLinie *>* CDBSchnittstelle::holenLinienplan ()

4.11.3.32 void CDBSchnittstelle::holenLinienplanBewertung (int *linienplanID*, int & *mittlereReisezeit*, int & *direktfahreranteil*)

Es werden die Werte mittlere Reisezeit und Direktfahreranteil fuer einen Linienplan aus der Tabelle LinienplanBewertung geholt

4.11.3.33 void CDBSchnittstelle::holenLinienplanModulParameter (int *aufruf*, int & *gesamtlaenge*, int & *maxLinienLaenge*, int & *maxLinienAnzahl*, double & *linienplanID*, int & *geschwindigkeit*, int & *algorithmus*, int & *anzeigegraphen*, bool & *EindeutigeLinienverkn*, bool & *MehrdeutigeLinienverkn*, bool & *Aufbruchsverknuepfung*, bool & *Linienendverknuepfung*, bool & *Basislinienwahl*)

4.11.3.34 CLinie * CDBSchnittstelle::holenLinienweg (int *id*)

4.11.3.35 h_array< int,three_tuple< int,int,bool >* >* CDBSchnittstelle::holenLinienwerte ()

Diese Methode holt alle Linienwerte aus der Datenbank

4.11.3.36 h_array< int,int >* CDBSchnittstelle::holenMindesthaltezeiten ()

4.11.3.37 h_array< int,list< three_tuple< int,int,int >* >*>* CDBSchnittstelle::holenMindestumsteigezeit ()

4.11.3.38 list< int >* CDBSchnittstelle::holenMoeglicheZielHaltestellen (int *input_id*)

4.11.3.39 list< CPendlerListeElement *>* CDBSchnittstelle::holenPendlerliste (void)

4.11.3.40 matrix * CDBSchnittstelle::holenPendlermatrix (void)

4.11.3.41 list< CPunkt *>* CDBSchnittstelle::holenPflichthaltestellen (void)

4.11.3.42 CLinie * CDBSchnittstelle::holenPflichtlinie (int *linienid*)

4.11.3.43 list< CLinie *>* CDBSchnittstelle::holenPflichtlinien ()

4.11.3.44 int CDBSchnittstelle::holenPhase ()

4.11.3.45 void CDBSchnittstelle::holenProjektInformationen (int *id*, char *& *beschreibung*, int & *phase*)

4.11.3.46 void CDBSchnittstelle::holenProjektInformationenPG369 (int *id*, char *& *beschreibung*, int & *phase*)

4.11.3.47 list< CQuelleZielListeElement *>* CDBSchnittstelle::holenQuelleZielListe (void)

4.11.3.48 matrix * CDBSchnittstelle::holenQuelleZielMatrix (void)

4.11.3.49 list< CQuelleZielListeElement *>* CDBSchnittstelle::holenUrsprungsQZListe ()

4.11.3.50 matrix * CDBSchnittstelle::holenUrsprungsQZMatrix ()

4.11.3.51 void CDBSchnittstelle::holenWegenetzKanten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0)

4.11.3.52 list< CWegenetzKnoten *>* CDBSchnittstelle::holenWegenetzKnoten ()

4.11.3.53 int CDBSchnittstelle::holenZusammenhangsKomponentenID (int *hst_eingabe*, int *Linie1_eingabe*)

4.11.3.54 array< list< three_tuple< int,int,int >* >* >* CDBSchnittstelle::holenZusammenhangskomponenten ()

Diese Methode holt alle Zusamenhangskomponenten aus der Datenbank

4.11.3.55 int CDBSchnittstelle::kopierenProjekt (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.11.3.56 int CDBSchnittstelle::kopierenProjektNachPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.11.3.57 int CDBSchnittstelle::kopierenProjektVonPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.11.3.58 void CDBSchnittstelle::loeschenAnschluss ()

Diese Methode loescht die Eintraege der Tabelle Anschluss

4.11.3.59 void CDBSchnittstelle::loeschenAusgangsloesung (int *loesungsID*)

Diese Methode loescht die Eintraege einer Ausgangsloesung aus der Tabelle Ausgangsloesung

4.11.3.60 void CDBSchnittstelle::loeschenAusgangsloesung-Haltestelle (int *loesungsID*)

Diese Methode loescht die Eintraege einer Ausgangsloesung aus der tabelle AusgangsloesungHaltestelle

4.11.3.61 void CDBSchnittstelle::loeschenFahrplan ()

Diese Methode loescht den fahrplan aus der Datenbank

4.11.3.62 void CDBSchnittstelle::loeschenKnotenWerte ()

Diese Methode loescht die Eintraege der Tabelle KnotenWerte

4.11.3.63 void CDBSchnittstelle::loeschenLinienplanBewertung ()

Es werden die Eintraege der Tabelle LinienplanBewertung gesloescht

4.11.3.64 void CDBSchnittstelle::loeschenLinienwerte ()

Diese Methode loescht die Eintraege der Tabelle LinienWerte

4.11.3.65 void CDBSchnittstelle::loeschenProjekt (int *id*)

4.11.3.66 void CDBSchnittstelle::loeschenProjektPG369 (int *id*)

4.11.3.67 void CDBSchnittstelle::loeschen-Zusammenhangskomponenten ()

Diese Methode loescht die Zusammenhangskomponenten

4.11.3.68 void CDBSchnittstelle::setzenAktuelleProjektID (int *id* = 0)

4.11.3.69 void CDBSchnittstelle::setztenPhase (int *phase*)

4.11.3.70 void CDBSchnittstelle::speichernAnschluss (list< three_tuple< int,int,int >* >* *anschluss*)

Diese Methode speichert alle direkten Anschluesse in der Datenbank

4.11.3.71 void CDBSchnittstelle::speichernAusgangsloesung (int *loesungsID*, double *wartezeitaufwand*, double *mindestwartezeit*, double *gesamtwartezeit*, int *anzahl_busse*)

Diese Methode speichert eine Ausgangsloesung in die Tabelle Ausgangsloesung die Werte sind LoesungsID, Wartezeitaufwand in Min, Mindestwartezeitaufwand in Min und Gesamtwartezeitaufwand in Min

4.11.3.72 void CDBSchnittstelle::speichernAusgangsloesung-Haltestelle (int, list< three_tuple< int,int,int >*>*)

Diese Methode speichert fuer eine Ausgangsloesung die Werte HaltestellenID, netzbedingte Wartezeit und Wartezeitaufwand

4.11.3.73 void CDBSchnittstelle::speichernAusschnitt (double *links*, double *oben*, double *rechts*, double *unten*)

4.11.3.74 void CDBSchnittstelle::speichernBevoelkerungsDichte (list< CBevoelkerungsdichteElement *>* *bevdichte* = 0)

4.11.3.75 void CDBSchnittstelle::speichernBusLinien (GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *fpgraph* = 0, list< CLinie *>* *buslinien* = 0, int *planid* = 0)

4.11.3.76 void CDBSchnittstelle::speichernFahrplan (list< CFahrplanElement *>* *fahrplan*)

Diese Methode speichert den Fahrplan in der Datenbank

4.11.3.77 void CDBSchnittstelle::speichernFahrplanModul-Parameter (int *maxfahrzeuganzahl*, int *allgemeinertakt*, int *mindest-haltezeit*, int *mindestumsteigezeit*)

4.11.3.78 void CDBSchnittstelle::speichernGuetemasse (CGuete-masse * *guetemasse*)

4.11.3.79 void CDBSchnittstelle::speichernHaltestellenModulParameter (int *aufruf*, int *algorithmus*, int *zufallsgeneratorInitialisierung*, double *minRadius*, double *maxRadius*, double *verbindungsRadiusStadt*, double *verbindungsRadiusLand*, int *radiustyp*, double *haltestellenLage*, double *toleranzgrenze*, int *bevoelkerungsRaster*, double *koeffizientBevoelkerungsueberversorgung*, double *koeffizientBevoelkerungsunterversorgung*, double *koeffizientFlaechenueberversorgung*, double *koeffizientFlaechenunterversorgung*, double *koeffizientFahrtwunscheinbeziehung*, double *koeffizientStrassenanpassung*, int *AnzahlHaltestellen*, bool *AnzStrassenkarte*, bool *AnzBevoelkerungsversorgung*, bool *AnzFlaechenversorgung*, bool *Anzkomplettunterdruecken*, int *Strategie*, int *AnzahlEltern*, int *AnzahlKinder*, int *Generationen*, bool *AbbruchGueteunterschreitung*, double *GrenzwGueteunterschreitung*)

4.11.3.80 void CDBSchnittstelle::speichernKantenStrasse (list< CPunkt *>* *arc* = 0, int *id* = 0)

4.11.3.81 void CDBSchnittstelle::speichernKnotenWerte (h_array< int,two_tuple< int,int >* >* *knotenwerte*)

Diese Methode speichert alle Knotenwerte in der Datenbank

4.11.3.82 void CDBSchnittstelle::speichernLinienplan (list< CLinie *>* *linienPlan* = 0)

4.11.3.83 void CDBSchnittstelle::speichernLinienplanBewertung (int *linienplanID*, int *mittlereReisezeit*, int *direktfahreranteil*)

Es werden die Werte Linienplan ID, mittlere Reisezeit und Direktfahreranteil in der Tabelle LinienplanBewertung gespeichert

4.11.3.84 void CDBSchnittstelle::speichernLinienplanModulParameter (int *aufruf*, int *gesamtlaenge*, int *maxLinienLaenge*, int *maxLinienAnzahl*, double *linienplanID*, int *geschwindigkeit* = 15, int *algorithmus* = 0, int *anzeigegraphen* = 0, bool *EindeutigeLinienverkn* = false, bool *MehrdeutigeLinienverkn* = false, bool *Aufbruchsverknuepfung* = false, bool *Linienendverknuepfung* = false, bool *Basislinienwahl* = false)

4.11.3.85 void CDBSchnittstelle::speichernLinienweg (CLinie * *linienweg* = 0)

4.11.3.86 void CDBSchnittstelle::speichernLinienwerte (h_array< int,three_tuple< int,int,bool >* >* *linienwerte*)

Diese Methode speichert alle Linenwerte in der Datenbank

4.11.3.87 void CDBSchnittstelle::speichernMindesthaltezeiten (h_array< int,int >* *mh*)

Speichern der Mindesthaltezeiten: Parameter bedeuten: HaltestelleID, Haltezeit (in Minuten)

4.11.3.88 void CDBSchnittstelle::speichernMindestumsteigezeit (h_array< int,list< three_tuple< int,int,int >* >*>* *input*)

Speichern die Mindestumsteigezeiten: Parameter bedeuten: HaltestellenID, Linienr1, Liniennr2, Mindestumsteigezeit

4.11.3.89 void CDBSchnittstelle::speichernPendlerliste (list< CPendlerListeElement *>* *pendlerliste* = 0)

4.11.3.90 void CDBSchnittstelle::speichernPendlermatrix (matrix * *pendlermatrix* = 0)

4.11.3.91 void CDBSchnittstelle::speichernPflichthaltestellen (list< CPunkt *>* *pflichthaltestellen* = NULL)

4.11.3.92 void CDBSchnittstelle::speichernPflichtlinien (list< CLinie *>* *pflichtlinien*)

4.11.3.93 void CDBSchnittstelle::speichernQuelleZielListe (list< CQuelleZielListeElement *>* *quellezielliste*)

4.11.3.94 void CDBSchnittstelle::speichernQuelleZielMatrix (matrix * *quellezielmatrix* = 0)

4.11.3.95 void CDBSchnittstelle::speichernUrsprungsQZListe (list< CQuelleZielListeElement *>* *quellezielliste*)

4.11.3.96 void CDBSchnittstelle::speichernUrsprungsQZMatrix (matrix * *quellezielmatrix* = 0)

4.11.3.97 void CDBSchnittstelle::speichernWegenetzKanten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0)

4.11.3.98 void CDBSchnittstelle::speichernWegenetzKnoten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetz-graph* = 0)

4.11.3.99 void CDBSchnittstelle::speichern-Zusammenhangskomponenten (array< list< three_tuple< int,int,int >* >* >* *zusammenhangskomponenten*)

Mit dieser Methode werden die Zusammenhangskomponenten in der datenbank gespeichert

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CDBSchnittstelle.h**
- **CDBSchnittstelle.cpp**

4.12 CEntscheidungsbaumverfahren Klassenreferenz

`#include <CEntscheidungsbaumverfahren.h>`

Klassendiagramm für CEntscheidungsbaumverfahren:

Öffentliche Datenelemente

- **CEntscheidungsbaumverfahren** (int,**CTransportkettengraph***,int)
- **~CEntscheidungsbaumverfahren** ()
- void **starten** ()
- void **stoppen** ()

4.12.1 Ausführliche Beschreibung

In dieser Klasse ist die Fahrplangenerierung per Entscheidungsbaumverfahren implementiert

4.12.2 Beschreibung der Konstruktoren und Destruktoren

4.12.2.1 CEntscheidungsbaumverfahren::CEntscheidungsbaumverfahren (int *zusammenhangskomponentenID*, CTransportkettengraph * *graph*, int *baumtiefe*)

Konstruktor des Entscheidungsbaumverfahrens. Der erste Wert gibt die ZusammenhangskomponentenID an

4.12.2.2 CEntscheidungsbaumverfahren::~CEntscheidungsbaumverfahren ()

Destruktor der Klasse CEntscheidungsbaumverfahren

4.12.3 Dokumentation der Elementfunktionen

4.12.3.1 void CEntscheidungsbaumverfahren::starten () [virtual]

Mit dieser Methode wird die Berechnung gestartet holen alle Kanten der Zusammenhangskomponente Berechne alle Kanten mit festerBindung Setzen Maximalgeruest Berechnen Maximalgeruest

Implementiert von **CLoesungsgenerator** (S. 117).

4.12.3.2 void CEntscheidungsbaumverfahren::stoppen () [virtual]

Mit dieser Methode wird die Berechnung kontrolliert gestoppt.

Implementiert von **CLoesungsgenerator** (S. 117).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CEntscheidungsbaumverfahren.h**
- **CEntscheidungsbaumverfahren.cpp**

4.13 CFahrplan Klassenreferenz

`#include <CFahrplan.h>`

Öffentliche Datenelemente

- **CFahrplan** (**CSchnittstellenTest***, GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>*)
- **CFahrplan** (**CFPSchnittstelle***, GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>*)
- **~CFahrplan** ()
- void **starten** ()
- void **stoppen** ()
- void **berechnenHeuristischeFahrplanerstellung** (int)
- void **berechnenSukzessiveLoesungsverbesserung** (int)
- void **berechnenEntscheidungsbaumverfahren** (int,int)
- void **ausgebenFahrplan** ()
- void **ausgebenWartezeiten** ()
- void **ausgebenFahrzeugAnzahl** ()
- void **ausgebenBewertung** ()

4.13.1 Ausführliche Beschreibung

Dies ist die Hauptklasse des Fahrplan - Moduls. Die komplette Kommunikation zwischen Programm - Modul und Fahrplan - Modul geschieht über diese Klasse. Desweiteren wird von dieser Klasse aus das Fahrplan - Modul gesteuert. Die Vorgehensweise wird in den einzelnen Methoden beschrieben.

4.13.2 Beschreibung der Konstruktoren und Destruktoren

4.13.2.1 CFahrplan::CFahrplan (CSchnittstellenTest * *Schnittstelle-Test*, GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *Netz*)

Wir erhalten von der Schnittstelle einen gerichteten Graphen. Dabei handelt es sich um einen parametrisierbaren LEDA Graphen.

An den Kanten wird als Information folgendes erwartet: 1. Verkehrsstrom 2. Fahrzeit 3. Liniennummer

An den Knoten sind folgende Werte zu finden: Haltestellen-ID

4.13.2.2 CFahrplan::CFahrplan (CFPSchnittstelle * *Schnittstelle*, GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *Netz*)

4.13.2.3 CFahrplan::~CFahrplan ()

Destruktor

4.13.3 Dokumentation der Elementfunktionen

4.13.3.1 void CFahrplan::ausgebenBewertung ()

Diese Methode benoetigt als Parameter eine "Matrix" in der fuer jedes Haltestellenpaar ein Object vom Typ **CReisebewertung** (S. 143) gespeichert ist.

4.13.3.2 void CFahrplan::ausgebenFahrplan ()

Diese Methode wird von der GUI aufgerufen, sobald der Benutzer sich einen Fahrplan erstellen lassen moechte. Dies geht erst, nachdem wir eine Loesung fuer wenigstens eine Zusammenhangskomponenten erstellt haben. Vorher ist der Button in der GUI Disabled. Dann werden ueber die Methode holenAbfahrtszeit die vom Benuzter eingegebenen Abfahrtszeiten fuer alle Zusammenhangskomponenten geholt. Dann werden ueber die Methode holenFahrplanIntervall, die Anfangs- und Endzeiten geholt, fuer die der Fahrplan generiert werden soll. Die beiden geholten Werte werden dann an die Methode berechnenFahrplan in **CTransportkettengraph** (S. 200) uebergeben. Dort wird dann mit Hilfe der **CZusammenhangskomponenten** (S. 215) eine liste von **CFahrplan-Element** (S. 58) (Datenstruktur) Objekten erzeugt. Diese Liste wird dann an die Programmgruppe uebergeben.

4.13.3.3 void CFahrplan::ausgebenFahrzeugAnzahl ()

Uebergabe der benoetigten FahrzeugAnzahl an die Schnittstelle

4.13.3.4 void CFahrplan::ausgebenWartezeiten ()

Uebergabe der Wartezeiten der berechneten Fahrplaene an die Schnittstelle

4.13.3.5 void CFahrplan::berechnenEntscheidungsbaumverfahren (int *zusammenhangskomponentenID*, int *baumtiefe*)

Diese Methode startet die Fahrplanberechnung per Entscheidungsbaumverfahren. Dazu wird als erstes die Mindesthaltezeiten und Mindestumsteigezeiten eingelesen. Dazu werden die Methoden holenMindestumsteigezeit und holenMindesthaltezeit aufgerufen. Als nächstes wird der Loesungsgenerator mit der Klasse **CEntscheidungsbaumverfahren** (S. 54) initialisiert, wobei die ID der zu berechneneden Zusammenhangskomponente uebergeben wird und der Fahrplan berechnet.

Parameter: 1. ID der Zusammenhangskomponente 2. Tiefe des Baumes (FOLGT SPAETER)

4.13.3.6 void CFahrplan::berechnenHeuristischeFahrplanerstellung (int *zusammenhangskomponentenID*)

Diese Methode startet die heuristische Fahrplanberechnung. Dazu wird als erstes die Mindesthaltezeiten und Mindestumsteigezeiten eingelesen. Dazu werden die Methoden holenMindestumsteigezeit und holenMindesthaltezeit aufgerufen. Dann wird die Methode holenFesteVerbindungen aufgerufen, die die vom Anwender gewaehlten Bindungen abruft, die keine netzbedingten Wartezeiten haben sollen. Gegebenenfalls wird eine Exception gefangen, die dann zum Abbruch der Berechnung und zur Rueckgabe einer Fehlermeldung an die Programmgruppe fuehrt. Weiterhin wird die Methode holenAllgemeinerTakt aufgerufen. Als nächstes wird der Loesungsgenerator mit der Klasse **CHeuristisches-Loesungsverfahren** (S. 97) initialisiert, wobei die ID der zu berechneneden Zusammenhangskomponente uebergeben und die Berechnung per Heuristik gestartet wird.

Parameter: 1. ID der Zusammenhangskomponente

4.13.3.7 void CFahrplan::berechnenSukzessive-Loesungsverbesserung (int *zusammenhangskomponentenID*)

Diese Methode startet die sukzessive Loesungsverbesserung. Dazu wird als erstes die Mindesthaltezeiten und Mindestumsteigezeiten eingelesen. Dazu werden die Methoden holenMindestumsteigezeit und holenMindesthaltezeit aufgerufen. Als nächstes wird der Loesungsgenerator mit der Klasse **CSukzessivVerfahren** (S. 178) initialisiert, wobei die ID der zu berechneneden Zusammenhangskomponente uebergeben wird und die gegebene Ausgangsloesung verbessert.

PRECONDITION: Ausgangsloesung liegt vor (Objekt Transportkettengraph)

Parameter: 1. ID der Zusammenhangskomponente

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! MUSS NOCH BEARBEITET WERDEN !!!

4.13.3.8 void CFahrplan::starten ()

Start - Methode, die vom Programmodul aufgerufen wird. Diese Methode erzeugt das Objekt Transportkettengraph der Klasse **CTransportkettengraph** (S. 200). An den Konstruktor der Klasse **CTransportkettengraph** (S. 200) uebergeben wir das Netz der Transportverbindungen. Nachdem alle Berechnungen abgeschlossen sind, uebergeben wir die Listen der Elemente der Zusammenhangskomponenten an die Programmgruppe (Funktion: berechnenListeDerHaltestellenInAllenZusammenhangskomponenten in **CTransportkettengraph** (S. 200)) und teilen ihr im Anschluss daran mit, das wir mit unseren Berechnungen fertig sind (Message). Zu diesem Zeitpunkt werden dann die entsprechenden Menueeintraege im Menu Berechnungen enabled.

4.13.3.9 void CFahrplan::stoppen ()

Stop - Methode des Fahrplan - Moduls. Mit dieser Methode wird das Fahrplan - Modul beendet.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! MUSS NOCH BEARBEITET WERDEN !!!

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFahrplan.h**
- **CFahrplan.cpp**

4.14 CFahrplanElement Klassenreferenz

`#include <CFahrplanElement.h>`

Öffentliche Datenelemente

- **CFahrplanElement** ()
- **CFahrplanElement** (int,int,int,int,int)
- **CFahrplanElement** (const CFahrplanElement&)
- CFahrplanElement **operator=** (const CFahrplanElement&)
- void **setzenHaltestelle** (int)
- void **setzenLinie** (int)
- void **setzenLinienStartHaltestelle** (int)
- void **setzenAbfahrtszeitStunde** (int)
- void **setzenAbfahrtszeitMinute** (int)
- int **gebenHaltestelle** ()
- int **gebenLinie** ()
- int **gebenLinienStartHaltestelle** ()

- int **gebenAbfahrtszeitStunde** ()
- int **gebenAbfahrtszeitMinute** ()

Freundbeziehungen

- istream& **operator>>** (istream&, CFahrplanElement&)
- ostream& **operator<<** (ostream&, const CFahrplanElement&)

4.14.1 Beschreibung der Konstruktoren und Destruktoren

4.14.1.1 CFahrplanElement::CFahrplanElement ()

Default Konstruktor

4.14.1.2 CFahrplanElement::CFahrplanElement (int *haltestellenID*, int *linienID*, int *startHaltestellenID*, int *stunde*, int *minute*)

Diesem Konstruktor wird ein Fahrplaneintrag uebergeben. 1. Parameter : HaltestellenID 2. Parameter : LinienID 3. Parameter : ID der Linien - Starthaltestelle 4. Parameter : Abfahrtszeit Stunde 5. Parameter : Abfahrtszeit Minute

4.14.1.3 CFahrplanElement::CFahrplanElement (const CFahrplanElement & *originalObjekt*)

Kopier Konstruktor

4.14.2 Dokumentation der Elementfunktionen

4.14.2.1 int CFahrplanElement::gebenAbfahrtszeitMinute ()

Geben-Methode des Attributes abfahrtszeitMinute Es wird die Minute der Abfahrtszeit zurueckgegeben

4.14.2.2 int CFahrplanElement::gebenAbfahrtszeitStunde ()

Geben-Methode des Attributes abfahrtszeitStunde Es wird die Stunde der Abfahrtszeit zurueckgegeben

4.14.2.3 int CFahrplanElement::gebenHaltestelle ()

Geben-Methode des Attributes haltestellenID Die ID wird zurueckgeliefert

4.14.2.4 int CFahrplanElement::gebenLinie ()

Geben-Methode des Attributes linienID Die ID der Linie wird zurueckgegeben

4.14.2.5 int CFahrplanElement::gebenLinienStartHaltestelle ()

Geben-Methode des Attributes startHaltestellenID Es wird die ID der Haltestelle zurueckgegeben, mit der die Linie beginnt. (Richtung der Linie)

4.14.2.6 CFahrplanElement CFahrplanElement::operator= (const CFahrplanElement & *originalObjekt*)

= Operator

4.14.2.7 void CFahrplanElement::setzenAbfahrtszeitMinute (int *minute*)

Setzen-Methode des Attributes AbfahrtszeitMinute Uebergeben wird die Minute der Abfahrtszeit

4.14.2.8 void CFahrplanElement::setzenAbfahrtszeitStunde (int *stunde*)

Setzen-Methode des Attributes Abfahrtszeittunde Uebergeben wird die Stunde der Abfahrtszeit

4.14.2.9 void CFahrplanElement::setzenHaltestelle (int *haltestellenID*)

Setzen-Methode des Attributes haltestellenID Uebergeben wird die HaltestellenID

4.14.2.10 void CFahrplanElement::setzenLinie (int *linienID*)

Setzen-Methode des Attributes linienID und startHaltestellenID Uebergeben wird die LinienID

4.14.2.11 void CFahrplanElement::setzenLinienStartHaltestelle (int *linienStartID*)

Setzen-Methode des Attributes startHaltestellenID Uebergeben wird die ID der Starthaltestelle

4.14.3 Freundbeziehungen und Funktionsdokumentation

4.14.3.1 ostream & operator<< (ostream & *os*, const CFahrplanElement & *originalFahrplan*) [friend]

<< Operator

4.14.3.2 istream & operator>> (istream & *is*, CFahrplanElement & *originalFahrplan*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFahrplanElement.h**
- **CFahrplanElement.cpp**

4.15 CFahrplanExceptions Klassenreferenz

`#include <Fehler.h>`

Klassendiagramm für CFahrplanExceptions:

Öffentliche Datenelemente

- **CFahrplanExceptions** ()
- **CFahrplanExceptions** (string)
- string **gebenMeldung** ()

4.15.1 Ausführliche Beschreibung

Default Exception der Fahrplangruppe, es wird im Konstruktor eine Meldung uebergeben

4.15.2 Beschreibung der Konstruktoren und Destruktoren

4.15.2.1 CFahrplanExceptions::CFahrplanExceptions () `[inline]`

4.15.2.2 CFahrplanExceptions::CFahrplanExceptions (string *meldung*)

Der Konstruktor enthaelt eine Meldung zur Fehler beschreibung

4.15.3 Dokumentation der Elementfunktionen

4.15.3.1 string CFahrplanExceptions::gebenMeldung ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.16 CFesteBindungNichtGefunden Klassenreferenz

`#include <Fehler.h>`

Klassendiagramm für CFesteBindungNichtGefunden:

Öffentliche Datenelemente

- **CFesteBindungNichtGefunden** (string)

4.16.1 Beschreibung der Konstruktoren und Destruktoren

4.16.1.1 CFesteBindungNichtGefunden::CFesteBindungNichtGefunden (string *meldung*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.17 CFortschritt Klassenreferenz

`#include <CFortschritt.h>`

Öffentliche Datenelemente

- **CFortschritt** (const char *titel="Vorgang wird bearbeitet...",bool batch=false)
- **~CFortschritt** ()
- void **anzeigenFortschritt** (unsigned long aktuell, unsigned long maximum)
- void **anzeigenFortschrittSteigenderAufwand** (unsigned long aktuell, unsigned long maximum)
- void **anzeigenAktivitaet** (void)
- void **meldenLaufzeitAusgabe** (char*)
- void **setzenTeilschritte** (unsigned int anzahl, char ** Teilschritte)
- void **setzenNaechstenSchritt** (void)
- void **setzenNaechstenSchritt** (unsigned int Schritt)
- unsigned long int **Summe** (unsigned long int Hoechstwert)

4.17.1 Beschreibung der Konstruktoren und Destruktoren

4.17.1.1 CFortschritt::CFortschritt (const char * *titel* = "Vorgang wird bearbeitet...", bool *batch* = false)

4.17.1.2 CFortschritt::~CFortschritt ()

4.17.2 Dokumentation der Elementfunktionen

4.17.2.1 unsigned long int CFortschritt::Summe (unsigned long int *Hoechstwert*)

Berechnet die Summe von 1 bis Hoechstwert.

4.17.2.2 void CFortschritt::anzeigenAktivitaet (void)

Meldet Programmaktivitaet in Form eines kleinen Rotators.

4.17.2.3 void CFortschritt::anzeigenFortschritt (unsigned long *aktuell*, unsigned long *maximum*)

Zeigt den Fortschritt des aktuellen Arbeitsschrittes an.

Parameter:

aktuell Aktuell erreichter Wert. Muss groesser oder gleich Null sein.

maximum Maximal zu erreichender Wert. Muss groesser als Null sein.

4.17.2.4 void CFortschritt::anzeigenFortschrittSteigenderAufwand (unsigned long *aktuell*, unsigned long *maximum*)

4.17.2.5 void CFortschritt::meldenLaufzeitAusgabe (char * *text*)

Gibt eine Meldung fuer den Benutzer aus. dient dem detaillierten kommentieren von Teilschritten.

4.17.2.6 void CFortschritt::setzenNaechstenSchritt (unsigned int *Schritt*)

4.17.2.7 void CFortschritt::setzenNaechstenSchritt (void)

Setzt den aktuellen Teuilschritt.

Die ParameterloseVariante aktiviert den naechsten Schritt, wohingegen die Parameterbehaftete den entsprechenden Array-Eintrag hervorhebt.

4.17.2.8 void CFortschritt::setzenTeilschritte (unsigned int *anzahl*, char ** *Teilschritte*)

Dient dem Aufnehmen von Teilschriiten, welche zur Visualisierung eines Gesamtfortschrittes herangezogen werden.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFortschritt.h**
- **CFortschritt.cpp**

4.18 CFPlanGraphKante Klassenreferenz

`#include <CFPlanGraphKante.h>`

Öffentliche Datenelemente

- **CFPlanGraphKante** ()
- **CFPlanGraphKante** (const CFPlanGraphKante&)
- CFPlanGraphKante **operator=** (const CFPlanGraphKante&)
- void **setzenLinienID** (int)
- void **setzenFahrtzeit** (int)
- void **setzenVerkehrsstrom** (int)
- int **holenLinienID** ()
- int **holenFahrtzeit** ()
- int **holenVerkehrsstrom** ()

Freundbeziehungen

- istream& **operator>>** (istream&, CFPlanGraphKante&)
- ostream& **operator<<** (ostream&, const CFPlanGraphKante&)

4.18.1 Beschreibung der Konstruktoren und Destruktoren

4.18.1.1 CFPlanGraphKante::CFPlanGraphKante ()

Default Konstruktor

4.18.1.2 CFPlanGraphKante::CFPlanGraphKante (const CFPlanGraphKante & *originalBeschriftung*)

Kopier Konstruktor

4.18.2 Dokumentation der Elementfunktionen

4.18.2.1 int CFPlanGraphKante::holenFahrtzeit ()

4.18.2.2 int CFPlanGraphKante::holenLinienID ()

4.18.2.3 int CFPlanGraphKante::holenVerkehrsstrom ()

4.18.2.4 CFPlanGraphKante CFPlanGraphKante::operator= (const CFPlanGraphKante & *originalBeschriftung*)

= Operator

4.18.2.5 void CFPlanGraphKante::setzenFahrtzeit (int *fahrtzeit*)

4.18.2.6 void CFPlanGraphKante::setzenLinienID (int *linienID*)

4.18.2.7 void CFPlanGraphKante::setzenVerkehrsstrom (int *strom*)

4.18.3 Freundbeziehungen und Funktionsdokumentation

4.18.3.1 ostream & operator<< (ostream & *os*, const CFPlanGraphKante & *beschriftung*) [friend]

<< Operator

4.18.3.2 istream & operator>> (istream & *is*, CFPlanGraphKante & *beschriftung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFPlanGraphKante.h**
- **CFPlanGraphKante.cpp**

4.19 CFPlanGraphKnoten Klassenreferenz

`#include <CFPlanGraphKnoten.h>`

Öffentliche Datenelemente

- **CFPlanGraphKnoten** ()
- **CFPlanGraphKnoten** (const CFPlanGraphKnoten&)
- CFPlanGraphKnoten **operator=** (const CFPlanGraphKnoten&)
- void **setzenID** (int)
- int **holenID** ()

Freundbeziehungen

- istream& **operator>>** (istream&, CFPlanGraphKnoten&)
- ostream& **operator<<** (ostream&, const CFPlanGraphKnoten&)

4.19.1 Beschreibung der Konstruktoren und Destruktoren

4.19.1.1 CFPlanGraphKnoten::CFPlanGraphKnoten ()

Default Konstruktor

4.19.1.2 CFPlanGraphKnoten::CFPlanGraphKnoten (const CFPlanGraphKnoten & *originalBeschriftung*)

Kopier Konstruktor

4.19.2 Dokumentation der Elementfunktionen

4.19.2.1 int CFPlanGraphKnoten::holenID ()

Holen-Methode des Attributes Knoten ID

4.19.2.2 CFPlanGraphKnoten CFPlanGraphKnoten::operator= (const CFPlanGraphKnoten & *originalBeschriftung*)

= Operator

4.19.2.3 void CFPlanGraphKnoten::setzenID (int *knotenID*)

Setzen-Methode des Attributes Knoten ID

4.19.3 Freundbeziehungen und Funktionsdokumentation

4.19.3.1 ostream & operator<< (ostream & *os*, const CFPlanGraphKnoten & *beschriftung*) [friend]

<< Operator

4.19.3.2 istream & operator>> (istream & *is*, CFPlanGraphKnoten & *beschriftung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFPlanGraphKnoten.h**
- **CFPlanGraphKnoten.cpp**

4.20 CFPSchnittstelle Klassenreferenz

`#include <CFPSchnittstelle.h>`

Klassendiagramm für CFPSchnittstelle:

Öffentliche Datenelemente

- **CFPSchnittstelle** (**CDatenbank**∗ datenbank, **CGUI**∗ gui)
- void **speichernGuetemasse** (**CGuetemasse** ∗)
- **CGuetemasse**∗ **holenGuetemasse** ()
- list<four_tuple<int,int,int,int>∗ >∗ **fahrplanerstellungDirekter-Anschluss** ()
- int **fahrplanerstellungMaximaleFahrzeuganzahl** ()
- int **fahrplanerstellungAllgemeinerTakt** ()
- void **fahrplanerstellungZusammenhangskomponenten** (array<list<three_tuple<int,int,int>∗ >∗ >∗)
- array<three_tuple<int,int,int>∗ >∗ **fahrplanerstellung-Abfahrtszeiten** ()
- void **fahrplanerstellungMenueaktivierung** ()
- h_array<int,int>∗ **fahrplanerstellungMindesthaltezeit** ()
- h_array<int,list<three_tuple<int,int,int>∗ >∗>∗ **fahrplanerstellung-Mindestumsteigezeit** ()
- four_tuple<int,int,int,int>∗ **fahrplanerstellungFahrplanIntervall** ()
- list<four_tuple<int, int, int, int>∗ >∗ **fahrplanerstellungFeste-Verbindungen** ()
- void **holenFahrplanModulParameter** (int& maxfahrzeuganzahl, int& allgemeinertakt, int& mindesthaltezeit, int& mindestumsteigezeit)
- void **fahrplanerstellungErgebnisrueckgabe** (list<**CFahrplan-Element**∗>∗)
- void **fahrplanerstellungWartezeiten** (double, double, double, double)
- void **fahrplanerstellungAnzahlBenoetigterFahrzeuge** (int)

4.20.1 Ausführliche Beschreibung

Diese Klasse bildet die Schnittstelle zwischen dem Fahrplanmodul, der GUI und der Datenbank. Sie leitet Anforderungen und Daten zwischen den anfragenden und den antwortenden Klassen weiter. Diese Klasse reagiert nur auf Anfrage des Fahrplanmoduls und arbeitet nicht selbststaendig.

4.20.2 Beschreibung der Konstruktoren und Destruktoren

4.20.2.1 CFPSchnittstelle::CFPSchnittstelle (CDatenbank * *datenbank*, CGUI * *gui*)

4.20.3 Dokumentation der Elementfunktionen

4.20.3.1 array< three_tuple< int,int,int >* >* CFPSchnittstelle::fahrplanerstellungAbfahrtszeiten ()

Die Methode liefert ein Array von four_tuple zurueck, welche folgende Werte haben:

- HaltestellenID
- Liniennr
- minute (Reicht aus, da in dem Intervall ein fester Takt gilt);

Das Array muss dabei die gleiche Reihenfolge haben, wie das Array der Zusammenhangskomponenten

4.20.3.2 int CFPSchnittstelle::fahrplanerstellungAllgemeinerTakt ()

Bei dieser Methode wird der Allgemeine Takt (Minuten - Wert) uebergeben.

4.20.3.3 void CFPSchnittstelle::fahrplanerstellungAnzahlBenoetigterFahrzeuge (int *fahrzeugAnzahl*)

Uebergabe der benoetigten Fahrzeug-Anzahl

4.20.3.4 list< four_tuple< int,int,int,int >* >* CFPSchnittstelle::fahrplanerstellungDirekterAnschluss ()

Diese Methode liefert in einer Liste alle tupel zurueck die eine feste Bindung haben sollen. Dabei wird die ZusammenhangskomponentenID, die ID der Linie 1, die ID der 2. Linie und die Umsteigehaltestellen ID uebergeben.

4.20.3.5 void CFPSchnittstelle::fahrplanerstellungErgebnisrueckgabe (list< CFahrplanElement *>* *fahrplan* = 0)

Diese Methode erhaelt den erstellten Fahrplan zur Speicherung in die Datenbank.

4.20.3.6 four_tuple< int,int,int,int >* CFPSchnittstelle::fahrplanerstellungFahrplanIntervall ()

Bei dieser Methode wird das Intervall zurueckgegeben, fuer das der Fahrplan gueltig sein soll Dabei gilt folgende Reihenfolge bei den int - Werten : Stunde Von, Minute Von, Stunde Bis, Minute Bis

4.20.3.7 list< four_tuple< int,int,int,int >* >* CFPSchnittstelle::fahrplanerstellungFesteVerbindungen ()

Setzen der Liste von festen Verbindungen

4.20.3.8 int CFPSchnittstelle::fahrplanerstellungMaximale-Fahrzeuganzahl ()

Hier soll die Maximale Fahrzeuganzahl uebergeben werden.

4.20.3.9 void CFPSchnittstelle::fahrplanerstellung-Menueaktivierung ()

Methode wird zum Aktivieren/Deaktivieren von Menueintraegen benoetigt

4.20.3.10 h_array< int,int >* CFPSchnittstelle::fahrplanerstellung-Mindesthaltezeit ()

Diese Methode uebergibt die Mindesthaltezeit fuer jede Haltestelle Die Daten werden in einem Hash_Array zurueckgegeben, indem der erste Wert die HaltestellenID und der zweite Wert die Haltezeit an dieser Haltestelle zurueckgegeben werden.

4.20.3.11 h_array< int,list< three_tuple< int,int,int >* >*>* CFPSchnittstelle::fahrplanerstellungMindestumsteigezeit ()

Diese Methode uebergibt die Mindestumsteigezeit fuer jede Kombination von Haltestelle und Liniennummern. Der Rueckgabewert baut sich wie folgt auf: Das erste Element im Hash_Array ist die HaltestellenID bei dem zweiten Element handelt es sich um eine Liste bestehend aus Liniennummer1, Liniennummer2 und die Mindestumsteigezeit zwischen diesen Linien.

4.20.3.12 void CFPSchnittstelle::fahrplanerstellungWartezeiten (double *summeNetzbedingteWartezeiten*, double *wartezeitAufwand*, double *mindestWartezeitAufwand*, double *gesamtWarteaufwand*)

Uebergabe der Ergebnisse der Fahrplanberechnung hinsichtlich der Wartezeiten 1. Summe der netzbedingten Wartezeiten 2. Wartezeit-Aufwand 3. Mindestwartezeit-Aufwand 4. Gesamt-Wartezeit-Aufwand

4.20.3.13 void CFPSchnittstelle::fahrplanerstellung-Zusammenhangskomponenten (array< list< three_tuple< int,int,int >* >* >* *zusammenhangskomponenten* = 0)

Diese Methode erhaelt die Zusammenhangskomponente Als Parameter wird ein Array der Zusammenhangskomponenten uebergeben, in dem wiederum eine Liste von three_tuple steht Die Werte sind dabei 1. HaltestellenID, 2. HaltestellenID und LinienID

4.20.3.14 void CFPSchnittstelle::holenFahrplanModulParameter (int & *maxfahrzeuganzahl*, int & *allgemeinertakt*, int & *mindesthaltezeit*, int & *mindestumsteigezeit*)

holen der Fahrplan Modul Parameter, die ueber die GUI eingegeben werden

4.20.3.15 CGuetemasse * CFPSchnittstelle::holenGuetemasse (void)

4.20.3.16 void CFPSchnittstelle::speichernGuetemasse (CGuetemasse * *guetemasse*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CFPSchnittstelle.h**
- **CFPSchnittstelle.cpp**

4.21 CGRASSSchnittstelle Klassenreferenz

`#include <CGRASSSchnittstelle.h>`

Öffentliche Datenelemente

- **CGRASSSchnittstelle** (bool bbetrieb=false)
- **~CGRASSSchnittstelle** ()
- list<list<**CPunkt***>*>* **holenStadtplan** (void)
- list<list<**CPunkt***>*>* **holenGebaeude** (void)
- list<list<**CPunkt***>*>* **holenWege** (void)
- list<**CPunkt***>* **holenHaltestellen** ()
- list<**CPunkt***>* **holenPflichthaltestellen** ()
- void **speichernAusschnittStrassennetz** (list<list<**CPunkt***>*>*)
- void **speichernHaltestellen** (list<**CPunkt***>*)
- void **speichernPflichthaltestellen** (list<**CPunkt***>*)
- void **speichernStrassenplan** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*)
- void **speichernLinienplan** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*, list<**CLinie***>*)
- void **speichernPflichtlinien** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*, list<**CLinie***>*)
- void **speichernEinePflichtlinie** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*, list<int>*, int)
- void **speichernRoute** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*, list<int>*)
- void **speichernZusammenhangskomponenten** (GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>*, array<list<three_tuple<int,int,int>* >* >*)

Geschützte Datenelemente

- int **speichernVectorlayer** (char* , list<list<**CPunkt***>*>*)
- int **speichernSiteslayer** (char*, list<**CPunkt***>*)
- list<**CPunkt***>* **holenSiteslayer** (char*)
- node* **holeNodezuHaltestellenID** (int id, GRAPH<**CWegenetzKnoten***, **CWegenetzKante***>* wegenetzgraph)

Geschützte Attribute

- bool **batchbetrieb**

4.21.1 Ausführliche Beschreibung

Stellt NUR die Verbindung von den Schnittstellen zu GRASS her. Alle anderen Zugriffe auf GRASS erfolgen direkt ueber die GUI.

4.21.2 Beschreibung der Konstruktoren und Destruktoren

4.21.2.1 CGRASSSchnittstelle::CGRASSSchnittstelle (bool *bbetrieb* = false)

Konstruktor, Destruktor

4.21.2.2 CGRASSSchnittstelle::~CGRASSSchnittstelle ()

4.21.3 Dokumentation der Elementfunktionen

4.21.3.1 node * CGRASSSchnittstelle::holeNodezuHaltestellenID (int *id*, GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0) [protected]

4.21.3.2 list< list< CPunkt *>*>* CGRASSSchnittstelle::holenGebaeude (void)

4.21.3.3 list< CPunkt *>* CGRASSSchnittstelle::holenHaltestellen ()

4.21.3.4 list< CPunkt *>* CGRASSSchnittstelle::holenPflichthaltestellen (void)

4.21.3.5 list< CPunkt *>* CGRASSSchnittstelle::holenSiteslayer (char * *dateiname*) [protected]

4.21.3.6 list< list< CPunkt *>*>* CGRASSSchnittstelle::holenStadtplan (void)

4.21.3.7 list< list< CPunkt *>*>* CGRASSSchnittstelle::holenWege (void)

4.21.3.8 void CGRASSSchnittstelle::speichernAusschnittStrassennetz (list< list< CPunkt *>*>* *listevonarcs* = 0)

die folgende Funktion ist dafuer zustaendig, die Testdaten bzw. die Ausgangsdaten des Haltestellen-Moduls in GRASS zu speichern

4.21.3.9 void CGRASSSchnittstelle::speichernEinePflichtlinie (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0, list< int >* *buslinie* = 0, int *nummer* = 0)

die folgende Funktion ist dafuer zustaendig, EINE von der GUI durchgereichte Pflichtlinie in GRASS zu speichern. Eingabe: den Wegenetzgraph um die Linienwege zu bekommen mit ihren Koordinaten, sowie eine Liste mit den Haltestellen der Pflicht-Buslinie nebst Nummer dieser Linie

4.21.3.10 void CGRASSSchnittstelle::speichernHaltestellen (list< CPunkt *>* *haltestellenliste*)

die folgende Funktion ist dafuer zustaendig, die vom Haltestellen-Modul zurueckgegebenen Haltestellen in GRASS

4.21.3.11 void CGRASSSchnittstelle::speichernLinienplan (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0, list< CLinie *>* *buslinien* = 0)

die folgende Funktion ist dafuer zustaendig, den vom Linienplan-Modul zurueckgegebenen Linienplan in GRASS zu speichern. Der Linienplan enthaellt alle Buslinien Eingabe: den Wegenetzgraph um die Linienwege zu bekommen mit ihren Koordinaten, sowie eine Liste mit den Buslinien

4.21.3.12 void CGRASSSchnittstelle::speichernPflichthaltestellen (list< CPunkt *>* *haltestellenliste*)

die folgende Funktion ist dafuer zustaendig, die Pflichthaltestellen aus der DB wieder auf den zugehoerigen Layer zu schieben

4.21.3.13 void CGRASSSchnittstelle::speichernPflichtlinien (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0, list< CLinie *>* *buslinien* = 0)

die folgende Funktion ist dafuer zustaendig, die von der GUI durchgereichten Pflichtlinien in GRASS zu speichern. Der Linienplan erhaellt dann alle diese

Pflichtlinien als Eingabe Eingabe: den Wegenetzgraph um die Linienwege zu bekommen mit ihren Koordinaten, sowie eine Liste mit den Pflicht-Buslinien

4.21.3.14 void CGRASSSchnittstelle::speichernRoute (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0, list< int >* *buslinie* = 0)

die folgende Funktion ist dafuer zustaendig, eine Route, die von der GUI kommt zu speichern in GRASS zu speichern. Eingabe: den Wegenetzgraph um die Linienwege zu bekommen mit ihren Koordinaten, sowie eine Liste mit den Haltestellen der Route

4.21.3.15 int CGRASSSchnittstelle::speichernSiteslayer (char * *dateiname*, list< CPunkt *>* *haltestellenliste*) [protected]

4.21.3.16 void CGRASSSchnittstelle::speichernStrassenplan (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph* = 0)

die folgende Funktion ist dafuer zustaendig, den vom Haltestellen-Modul zurueckgegebenen Strassenplan in GRASS zu speichern. Der Strassenplan enthaellt die Strassen, ueber die die Busse spaeter fahren koennen.

4.21.3.17 int CGRASSSchnittstelle::speichernVectorlayer (char * *dateiname*, list< list< CPunkt *>*>* *listevonarcs* = 0) [protected]

4.21.3.18 void CGRASSSchnittstelle::speichern-Zusammenhangskomponenten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph*, array< list< three_tuple< int,int,int >* >* >* *zusammenhangskomponenten*)

die folgende Funktion ist dafuer Zustaendig, die vom Fahrplan zurueck gegebenen Zusammenhangskomponenten anzuzeigen Eingabe: der Wegenetzgraph, um die Koordinaten heraus zu finden Ein Array, in dem in jeder Liste eine Zusammenhangskomponente liegt.

4.21.4 Dokumentation der Datenelemente

4.21.4.1 bool CGRASSSchnittstelle::batchbetrieb [protected]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CGRASSSchnittstelle.h**
- **CGRASSSchnittstelle.cpp**

4.22 CGuetemasse Klassenreferenz

`#include <CGuetemasse.h>`

Öffentliche Datenelemente

- **CGuetemasse** ()
- **~CGuetemasse** ()
- double **holenHaltestellenBewertung** ()
- void **speichernAnzahlHaltestellen** (int)
- int **holenAnzahlHaltestellen** ()
- void **speichernBevoelkerungsUeberversorgung** (double)
- double **holenBevoelkerungsUeberversorgung** ()
- void **speichernBevoelkerungsUnterversorgung** (double)
- double **holenBevoelkerungsUnterversorgung** ()
- void **speichernFlaechenUeberversorgung** (double)
- double **holenFlaechenUeberversorgung** ()
- void **speichernFlaechenUnterversorgung** (double)
- double **holenFlaechenUnterversorgung** ()
- void **speichernFlaechenVersorgunginProzent** (double)
- double **holenFlaechenVersorgunginProzent** ()
- void **speichernBevoelkerungsVersorgunginProzent** (double)
- double **holenBevoelkerungsVersorgunginProzent** ()
- void **speichernFusswegProHaltestelle** (double)
- double **holenFusswegProHaltestelle** (void)
- void **speichernReisezeitHS2HSAuto** (map2<int, int, double > *)
- void **setzenReisezeitHS2HSAuto** (map2<int, int, double > *)
- map2<int, int, double >* **holenReisezeitHS2HSAuto** (void)
- void **setzenQuelleZielMatrix** (matrix *)
- void **speichernAnzahlUmstiege** (int anzUmstiege)
- int **holenAnzahlUmstiege** ()
- void **speichernAnzahlVerbindungenmitUmstiegzwang** (double verbindungenmitUmstiegezwang)
- double **holenAnzahlVerbindungenmitUmstiegzwang** ()
- void **speichernGesamtfahrer** (int gesFahrer)
- int **holenGesamtfahrer** ()
- void **speichernDirektfahrer** (int dirFahrer)
- int **holenDirektfahrer** ()
- void **speichernMittlereReisezeit** (int mittlereReisezt)
- int **holenMittlereResezeit** ()
- void **speichernProzentsatzVerbindungendiekeineDirektfahrten** (double prozentVerbdiekeineDirektfahrten)
- double **holenProzentsatzVerbindungendiekeineDirektfahrten** ()
- void **speichernDirektfahreranteil** (double direktFahrerAnt)
- double **holenDirektfahreranteil** ()
- void **speichernStreckenkilometer** (int streckenkm)

- int **holenStreckenkilometer** ()
- void **speichernGesamteStreckenkilometer** (int gesamteStreckenkm)
- int **holenGesamteStreckenkilometer** ()
- void **speichernMittlereMindestreisezeit** (int mittleremindestreisezt)
- int **holenMittlereMindestreisezeit** ()
- void **speichernReisezeitverlaengerung** (double reiseztverlaengerung)
- double **holenReisezeitverlaengerung** ()
- void **setzenLinienanzahl** (int)
- int **holenLinienanzahl** (void)
- void **speichernMittlerenVerknuepfungsgrad** (double)
- double **holenMittlerenVerknuepfungsgrad** (void)
- void **speichernAnzahlBusse** (double)
- double **holenAnzahlBusse** (void)
- void **speichernNetzbedingteWartezeitenGesamt** (double)
- double **holenNetzbedingteWartezeitenGesamt** (void)
- void **speichernWartezeiteAufwandGesamt** (double)
- double **holenWartezeitenAufwandGesamt** (void)
- void **speichernMindestwartezeitaufwand** (double)
- double **holenMindestwartezeitaufwand** (void)
- void **speichernGesamtzeitaufwand** (double)
- double **holenGesamtzeitaufwand** (void)
- void **speichernFahrplanwirkungsgrad** (double)
- double **holenFahrplanwirkungsgrad** (void)
- void **speichernFahrgastfahrten** (int)
- int **holenFahrgastfahrten** (void)
- void **setzenFahrplanintervall** (double)
- void **speichernDurchschnittlicheAnzahlBusse** (double)
- double **holenDurchschnittlicheAnzahlBusse** (void)
- void **speichernMittlereReisezeit** (double)
- double **holenMittlereReisezeit** (void)
- void **speichernAnzahlPersonen** (double)
- double **holenAnzahlPersonen** (void)
- void **speichernReisezeitZuDirektfahrer** (double)
- double **holenReisezeitZuDirektfahrer** (void)
- void **speichernMindestreisezeitZuDirektfahrer** (double)
- double **holenMindestreisezeitZuDirektfahrer** (void)
- void **erzeugenNetzbedingtenWartezeitenArray** (void)
- void **setzenNetzbedingteWartezeiten** (h_array< three_tuple<int, int, int>*, int> *)
- h_array< three_tuple<int, int , int>*, int>* **holenNetzbedingte-Wartezeiten** (void)
- map2<int, int, **CReisebewertung** >* **holenReisebewertung** (void)
- void **speichernReisebewertung** (map2 <int, int, **CReisebewertung** > *)

4.22.1 Ausführliche Beschreibung

In dieser Klasse werden die Guetemasse aller Module abgelegt.

4.22.2 Beschreibung der Konstruktoren und Destruktoren

4.22.2.1 CGuetemasse::CGuetemasse ()

Standardkonstruktor.

4.22.2.2 CGuetemasse::~CGuetemasse ()

Destruktor

4.22.3 Dokumentation der Elementfunktionen

4.22.3.1 void CGuetemasse::erzeugenNetzbedingtenWartezeiten-Array (void)

Erzeugen eines HashArrays fuer die Netzbedingten Wartezeiten

4.22.3.2 double CGuetemasse::holenAnzahlBusse (void)

4.22.3.3 int CGuetemasse::holenAnzahlHaltestellen ()

4.22.3.4 double CGuetemasse::holenAnzahlPersonen (void)

Gibt die durchschnittliche Anzahl Personen pro Bus in einem Takt zurueck

4.22.3.5 int CGuetemasse::holenAnzahlUmstiege ()

4.22.3.6 double CGuetemasse::holenAnzahlVerbindungenmit-Umstiegzwang ()

4.22.3.7 double CGuetemasse::holenBevoelkerungs-Ueberversorgung ()

4.22.3.8 double CGuetemasse::holenBevoelkerungsUnterversorgung ()

4.22.3.9 double CGuetemasse::holenBevoelkerungsVersorgungin-Prozent ()

4.22.3.10 int CGuetemasse::holenDirektfahrer ()

4.22.3.11 double CGuetemasse::holenDirektfahreranteil ()

4.22.3.12 double CGuetemasse::holenDurchschnittlicheAnzahlBusse (void)

Gibt die durchschnittliche Anzahl an Bussen pro Linie zurueck

4.22.3.13 int CGuetemasse::holenFahrgastfahrten (void)

4.22.3.14 double CGuetemasse::holenFahrplanwirkungsgrad (void)

4.22.3.15 double CGuetemasse::holenFlaechenUeberversorgung ()

4.22.3.16 double CGuetemasse::holenFlaechenUnterversorgung ()

4.22.3.17 double CGuetemasse::holenFlaechenVersorgunginProzent ()

4.22.3.18 double CGuetemasse::holenFusswegProHaltestelle (void)

4.22.3.19 int CGuetemasse::holenGesamteStreckenkilometer ()

4.22.3.20 int CGuetemasse::holenGesamtfahrer ()

4.22.3.21 double CGuetemasse::holenGesamtzeitaufwand (void)

4.22.3.22 double CGuetemasse::holenHaltestellenBewertung ()

Liest die Gesamtbewertung. Dabei ist sie die Summe aus den Einzelbewertungen.

4.22.3.23 int CGuetemasse::holenLinienanzahl (void)

4.22.3.24 double CGuetemasse::holenMindestreisezeitZuDirektfahrer (void)

Verhaeltnis der mittleren Mindestreisezeit im Vergleich zur Direktfahrt (mit Auto) Das Ergebnis ist als Prozentwert zu interpretieren

4.22.3.25 double CGuetemasse::holenMindestwartezeitaufwand (void)

4.22.3.26 int CGuetemasse::holenMittlereMindestreisezeit ()

4.22.3.27 double CGuetemasse::holenMittlereReisezeit (void)

Gibt die mittlere Reisezeit (Fussweg, Fahrtreisezeit, Wartezeit) zurueck

4.22.3.28 int CGuetemasse::holenMittlereResezeit ()

4.22.3.29 double CGuetemasse::holenMittlerenVerknuepfungsgrad (void)

4.22.3.30 h_array< three_tuple< int,int,int >*,int >* CGuetemasse::holenNetzbedingteWartezeiten (void)

Gibt die Netzbedingten Wartezeiten zurueck

4.22.3.31 double CGuetemasse::holenNetzbedingteWartezeiten-Gesamt (void)

4.22.3.32 double CGuetemasse::holenProzentsatz-VerbindungendiekeineDirektfahrten ()

4.22.3.33 map2< int,int,CReisebewertung >* CGuetemasse::holen-Reisebewertung (void)

Gibt die map2 Datenstruktur mit zurueck

4.22.3.34 map2< int,int,double >* CGuetemasse::holenReisezeit-HS2HSAuto (void)

Methode, mit dem das Haltestellen-Modul die Entfernung in Metern von jeder Haltestelle zu jeder anderen setzt. Dazu wird eine Durchschnittsgeschwindigkeit eines Autos von 28 km/h angenommen, was 466,67 zurueckgelegten Metern in einer Minute entspricht

4.22.3.35 double CGuetemasse::holenReisezeitZuDirektfahrer (void)

Verhaeltnis der mittleren Reisezeit im Vergleich zur Direktfahrt (mit Auto) Das Ergebnis ist als Prozentwert zu interpretieren

4.22.3.36 double CGuetemasse::holenReisezeitverlaengerung ()

4.22.3.37 int CGuetemasse::holenStreckenkilometer ()

4.22.3.38 double CGuetemasse::holenWartezeitenAufwandGesamt (void)

4.22.3.39 void CGuetemasse::setzenFahrplanintervall (double *fpi*)

Setzt das Fahrplaninterval

4.22.3.40 void CGuetemasse::setzenLinienanzahl (int *anzLinien*)

Anzahl der Linien insgesamt

4.22.3.41 void CGuetemasse::setzenNetzbedingteWartezeiten (h_array< three_tuple< int,int,int >*,int >* *NWArray*)

Setzen der Netzbedingten Wartezeiten

4.22.3.42 void CGuetemasse::setzenQuelleZielMatrix (matrix * *qzMatrix*)

Setzen-Methode fuer die Quelle-Ziel-Matrix. Diese wird fuer die Berechnung der Personenanzahl pro Takt benoetigt Diese Methode wird von den Haltestellen aufgerufen, sodass die Quelle Ziel-Matrix die aktuell von den Haltestellen berechneten Werte enthaelt.

4.22.3.43 void CGuetemasse::setzenReisezeitHS2HSAuto (map2< int,int,double >* *reisezeit_Entfernung*)

diese Methode wird vom Haltestellen-Modul benoetigt. Es werden die ENTFERNUNGEN von jeder Haltestelle zu jeder anderen gesetzt. Die Entfernungen sind die kuerzesten Wege. Diese Methode berechnet daraus die Reisezeit von jeder Haltestelle zu jeder anderen.

4.22.3.44 void CGuetemasse::speichernAnzahlBusse (double *anzBusse*)

4.22.3.45 void CGuetemasse::speichernAnzahlHaltestellen (int *anzHaltestellen*)

4.22.3.46 void CGuetemasse::speichernAnzahlPersonen (double *anzPersonen*)

Setzt die durchschnittliche Anzahl Personen pro Bus in einem Takt

4.22.3.47 void CGuetemasse::speichernAnzahlUmstiege (int *anzUmstiege*)

Die Anzahl der Umstiege im gesamten Linienplan

4.22.3.48 void CGuetemasse::speichernAnzahlVerbindungenmit-Umstiegzwang (double *verbindungenmitUmstiegezwang*)

durchgschnittliche Anzahl der Umstiege

4.22.3.49 void CGuetemasse::speichernBevoelkerungs-Ueberversorgung (double *bevUeberversogung*)

4.22.3.50 void CGuetemasse::speichernBevoelkerungs-Unterversorgung (double *bevUnterversorgung*)

4.22.3.51 void CGuetemasse::speichernBevoelkerungsVersorgungin-Prozent (double *bevVersorgunginProzent*)

4.22.3.52 void CGuetemasse::speichernDirektfahrer (int *dirFahrer*)

die Anzahl der Direktfahrer

4.22.3.53 void CGuetemasse::speichernDirektfahreranteil (double *direktFahrerAnt*)

Der Anteil der Direktfahrer unter allen Fahrern

4.22.3.54 void CGuetemasse::speichernDurchschnittlicheAnzahl-Busse (double *anzBusse*)

Setzt die durchschnittliche Anzahl an Bussen pro Linie

4.22.3.55 void CGuetemasse::speichernFahrgastfahrten (int *neu-Fahrgastfahrten*)

4.22.3.56 void CGuetemasse::speichernFahrplanwirkungsgrad (double *wirkungsgrad*)

4.22.3.57 void CGuetemasse::speichernFlaechenUeberversorgung (double *flUeberversogung*)

4.22.3.58 void CGuetemasse::speichernFlaechenUnterversorgung (double *flUnterversorgung*)

4.22.3.59 void CGuetemasse::speichernFlaechenVersorgungin-Prozent (double *flVersorgunginProzent*)

4.22.3.60 void CGuetemasse::speichernFusswegProHaltestelle (double *fWeg*)

Der durchschnittliche Fussweg pro Haltestelle

4.22.3.61 void CGuetemasse::speichernGesamteStreckenkilometer (int *gesamteStreckenkm*)

Die Gesamtstreckenkilometer

4.22.3.62 void CGuetemasse::speichernGesamtfahrer (int *gesFahrer*)

die Gesamtfahrer

4.22.3.63 void CGuetemasse::speichernGesamtzeitaufwand (double *gesZeitaufwand*)

4.22.3.64 void CGuetemasse::speichernMindestreisezeitZuDirektfahrer (double *mrzZuDf*)

Verhaeltnis der mittleren Mindestreisezeit im Vergleich zur Direktfahrt (mit Auto) Das Ergebnis ist als Prozentwert zu interpretieren

4.22.3.65 void CGuetemasse::speichernMindestwartezeitaufwand (double *mindwartezeit*)

4.22.3.66 void CGuetemasse::speichernMittlereMindestreisezeit (int *mittleremindestreisezt*)

Die mittlere Mindestreisezeit

4.22.3.67 void CGuetemasse::speichernMittlereReisezeit (double *mittlereRZGlobal*)

Setzt die mittlere Reisezeit (Fussweg, Mindestreisezeit, Wartezeit)

4.22.3.68 void CGuetemasse::speichernMittlereReisezeit (int *mittlereReisezt*)

die mittlere Reisezeit

4.22.3.69 void CGuetemasse::speichernMittlerenVerknuepfungsgrad (double *grad*)

Der mittlere Verknuepfungsgrad

4.22.3.70 void CGuetemasse::speichernNetzbedingteWartezeiten-Gesamt (double *netzWartezeit*)

4.22.3.71 void CGuetemasse::speichernProzentsatz-VerbindungendiekeineDirektfahrten (double *prozentVerbdiekeine-Direktfahrten*)

Anteil der Verbingungen mit Umstiegszwang

4.22.3.72 void CGuetemasse::speichernReisebewertung (map2< int,int,CReisebewertung >* *neueReisebewertung*)

Speichert die map2 Datenstruktur mit zurueck

4.22.3.73 void CGuetemasse::speichernReisezeitHS2HSAuto (map2< int,int,double >* *reisezeit*)

die direkte Reisezeit von Haltestelle zu Haltestelle (mit dem Auto) Methode fuer das Programm

4.22.3.74 void CGuetemasse::speichernReisezeitZuDirektfahrer (double *rzZudf*)

Verhaeltnis der mittleren Reisezeit im Vergleich zur Direktfahrt (mit Auto) Das Ergebnis ist als Prozentwert zu interpretieren

4.22.3.75 void CGuetemasse::speichernReisezeitverlaengerung (double *reiseztverlaengerung*)

Die Reisezeitverlaengerung

4.22.3.76 void CGuetemasse::speichernStreckenkilometer (int *streckenkm*)

Die Streckenkilometer des Liniennetzes

4.22.3.77 void CGuetemasse::speichernWartezeiteAufwandGesamt (double *wartezeit*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CGuetemasse.h**
- **CGuetemasse.cpp**

4.23 CGuetemasseFehler Klassenreferenz

`#include <Fehler.h>`

Öffentliche Datenelemente

- **CGuetemasseFehler** ()
- **CGuetemasseFehler** (string)
- string **ausgebenMeldung** (void)

4.23.1 Beschreibung der Konstruktoren und Destruktoren

4.23.1.1 CGuetemasseFehler::CGuetemasseFehler () `[inline]`

4.23.1.2 CGuetemasseFehler::CGuetemasseFehler (string *fehlerMeldung*)

4.23.2 Dokumentation der Elementfunktionen

4.23.2.1 string CGuetemasseFehler::ausgebenMeldung (void)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.24 CGUI Klassenreferenz

`#include <CGUI.h>`

Öffentliche Datenelemente

- **CGUI** (int argc=0, char ** argv=0)
- **~CGUI** ()
- void **laufen** ()
- void **abfragenAusgangsloesung** ()
- list<four_tuple<int,int,int,int>* >* **direkterAnschluss** ()
- int **maximaleFahrzeuganzahl** ()
- int **allgemeinerTakt** ()
- array<three_tuple<int,int,int>* >* **abfahrtszeiten** ()
- h_array<int,int>* **mindesthaltezeit** ()
- h_array<int,list<three_tuple<int,int,int>* >*>* **mindestumsteigezeit** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- void **aktivierenMenue** ()
- void **ausgebenMeldung** (char*)
- void **ausgebenLaufzeitmeldung** (char*)
- void **beendenHaltestellenberechnung** (int,char* s=0)
- void **beendenLinienplanberechnung** (int,char* s=0)
- void **beendenFahrplaninit** (int,char* s=0)

- void **beendenFahrplanberechnung** (int,char* s=0)
- void **beendenFahrplanfertigstellen** (int,char* s=0)
- void **beendenFahrplanausgebenBewertung** (int,char* s=0)
- void **holenDaten** (matrix*&, list<**CQuelleZielListeElement***>*&)
- void **holenDaten** (matrix*&, list<**CPendlerListeElement***>*&)

4.24.1 Ausführliche Beschreibung

stellt die grafische Oberflaeche dar und reicht Benutzereingaben weiter an

4.24.2 Beschreibung der Konstruktoren und Destruktoren

4.24.2.1 CGUI::CGUI (int *argc* = 0, char ** *argv* = 0)

Konstruktor legt ein neues Kontroll-Objekt an, das seinerseits wiederum die Schnittstellen erzeugt (vgl. Konstruktor von **CKontrolle** (S. 107) in **CKontrolle.h**) Dem Kontrollmodul muss dabei ein Zeiger auf das GUI-Objekt selbst (this) uebergeben werden. Dieses reicht den Zeiger an die Schnittstellen weiter

4.24.2.2 CGUI::~CGUI ()

Destruktor dient zum Aufraeumen, bevor das Programm terminiert

4.24.3 Dokumentation der Elementfunktionen

4.24.3.1 array< three_tuple< int,int,int >* >* CGUI::abfahrtszeiten ()

abfahrtszeiten wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach den Abfahrtszeiten Werte der Tupel dabei:

- HaltestellenID
- Liniennr
- minute (Reicht aus, da in dem Intervall ein fester Takt gilt);

4.24.3.2 void CGUI::abfragenAusgangsloesung ()

abfrageAusgangsloesung zeigt die vom Fahrplanmodul generierten Ausgangsloesungen dem Benutzer an und fordert diesen auf, sich fuer eine dieser Ausgangsloesungen zu entscheiden. Die vom Benutzer gewaehlte Loesung wird dann an das Fahrplanmodul zurueckgeliefert (Rueckgabewert).

4.24.3.3 void CGUI::aktivierenMenue ()

Aktiviert bzw. deaktiviert einen entsprechenden Menuepunkt

4.24.3.4 int CGUI::allgemeinerTakt ()

allgemeinerTakt wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach dem allgemeinen Takt

4.24.3.5 void CGUI::ausgebenLaufzeitmeldung (char * *s*)

4.24.3.6 void CGUI::ausgebenMeldung (char * *s*)

Die folgenen Funktionen dienen zur Ausgabe, der von den Modulen uebergebenen Texte

4.24.3.7 void CGUI::beendenFahrplanausgebenBewertung (int *i*, char * *s* = 0)

4.24.3.8 void CGUI::beendenFahrplanberechnung (int *i*, char * *s* = 0)

4.24.3.9 void CGUI::beendenFahrplanfertigstellen (int *i*, char * *s* = 0)

4.24.3.10 void CGUI::beendenFahrplaninit (int *i*, char * *s* = 0)

4.24.3.11 void CGUI::beendenHaltestellenberechnung (int *i*, char * *s* = 0)

Die folgenden Funktionen dienen dazu, der GUI mitzuteilen, dass und wie das entsprechende Modul seine Berechnungen beendet hat. Die GUI kann daraufhin seine Anzeige aktuallisieren

4.24.3.12 void CGUI::beendenLinienplanberechnung (int *i*, char * *s* = 0)

4.24.3.13 list< four_tuple< int,int,int,int >* >* CGUI::direkter-Anschluss ()

direkterAnschluss wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach den direkten Anschluessen four_tuple (= ein Anschluss) hat die folgende Werte: ZusammenhangskomponentenID, die ID der Linie 1, die ID der 2. Linie und die Umsteigehaltestellen ID

4.24.3.14 four_tuple< int,int,int,int >* CGUI::fahrplanIntervall ()

fahrplanintervall wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach den fahrplanintervallen Dabei gilt folgende Reihenfolge bei den int - Werten : Stunde Von, Minute Von, Stunde Bis, Minute Bis

4.24.3.15 void CGUI::holenDaten (matrix *& *pendlerMatrix*, list< CPendlerListeElement *>*& *pendlerListe*)

Hole Daten fuer die Haltestellen PendlerMatrix und PendlerListe

4.24.3.16 void CGUI::holenDaten (matrix *& *quelleZielMatrix*, list< CQuelleZielListeElement *>*& *quelleZielListe*)

Hole Daten fuer die Haltestellen QuelleZielMatrix und QuelleZielListe

4.24.3.17 void CGUI::laufen ()

Diese Funktion startet die Ereigniswarteschleife. Wird beim Programmstart benoetigt: die main-Funktion erzeugt ein Objekt GUI. Anschliessend ruft die main-Funktion diese Funktion laufe() auf, mit der die Ereigniswarteschleife gestartet wird. Sobald diese Funktion verlassen wird, Terminiert das Programm. Entspricht der sonst ueblichen "Run" Funktion

4.24.3.18 int CGUI::maximaleFahrzeuganzahl ()

maximaleFahrzeuganzahl wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach der maximal verfuegabren Fahrzeuganzahl

4.24.3.19 h_array< int,int >* CGUI::mindesthaltezeit ()

mindesthaltezeit wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach den mindesthaltezeit Die Daten werden in einem Hash_Array zurueckgegeben, indem der erste Wert die HaltestellenID und der zweite Wert die Haltezeit an dieser Haltestelle zurueckgegeben werden.

4.24.3.20 h_array< int,list< three_tuple< int,int,int >* >*>* CGUI::mindestumsteigezeit ()

mindestumsteigezeit wird vom Farhplanmodul aufgerufen. fragt den Benutzer nach den mindestumsteigezeiten Der Rueckgabewert baut sich wie folgt auf: Das erste Element im Hash_Array ist die HaltestellenID bei dem zweiten Element handelt es sich um eine Liste bestehend aus Liniennummer1, Liniennummer2 und die Mindestumsteigezeit zwischen diesen Linien.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CGUI.h**
- **CGUI.cpp**

4.25 CGUISchnittstelle Klassenreferenz

`#include <CGUISchnittstelle.h>`

Klassendiagramm für CGUISchnittstelle:

Öffentliche Datenelemente

- **CGUISchnittstelle** (**CDatenbank***)
- **CGuetemasse*** **holenGuetemasse** ()
- void **speichernGuetemasse** (**CGuetemasse*** guetemasse)
- void **speichernHaltestellenModulParameter** (int aufruf, int algorithmus, int zufallsgeneratorInitialisierung, double minRadius, double maxRadius, double verbindungsRadiusStadt, double verbindungsRadiusLand, int radiustyp, double haltestellenLage, double toleranzgrenze, int bevoelkerungsRaster, double koeffizientBevoelkerungsueberversorgung, double koeffizientBevoelkerungsunterversorgung, double koeffizientFlaechenueberversorgung, double koeffizientFlaechenunterversorgung, double koeffizientFahrtwunscheinbeziehung, double koeffizientStrassenanpassung, int AnzahlHaltestellen, bool AnzStrassenkarte, bool AnzBevoelkerungsversorgung, bool AnzFlaechenversorgung, bool Anzkomplettunterdruecken, int Strategie, int AnzahlEltern, int AnzahlKinder, int Generationen, bool AbbruchGueteunterschreitung, double GrenzwGueteunterschreitung)
- void **holenHaltestellenModulParameter** (int aufruf, int& algorithmus, int& zufallsgeneratorInitialisierung, double& minRadius, double& maxRadius, double& verbindungsRadiusStadt, double& verbindungsRadiusLand, int& radiustyp, double& haltestellenLage, double& toleranzgrenze, int& bevoelkerungsRaster, double& koeffizientBevoelkerungsueberversorgung, double& koeffizientBevoelkerungsunterversorgung, double& koeffizientFlaechenueberversorgung, double& koeffizientFlaechenunterversorgung, double& koeffizientFahrtwunscheinbeziehung, double& koeffizientStrassenanpassung, int& AnzahlHaltestellen, bool& AnzStrassenkarte, bool& AnzBevoelkerungsversorgung, bool& AnzFlaechenversorgung, bool& Anzkomplettunterdruecken, int& Strategie, int& AnzahlEltern, int& AnzahlKinder, int& Generationen, bool& AbbruchGueteunterschreitung, double& GrenzwGueteunterschreitung)
- void **speichernBevoelkerungsDichte** (list<**CBevoelkerungsdichteElement***>*)
- void **speichernUrsprungsQZMatrix** (matrix* quelleZielMatrix, list<**CQuelleZielListeElement***>* quelleZielListe)
- void **speichernPendlermatrix** (matrix* pendlerMatrix, list<**CPendlerListeElement***>* pendlerListe)
- void **speichernLinienplanModulParameter** (int aufruf, int gesamtlaenge, int maxLinienLaenge, int maxLinienAnzahl, double linienplanID, int geschwindigkeit=15, int algorithmus=0, int anzeigegraphen=0, bool EindeutigeLinienverkn=false, bool MehrdeutigeLinienverkn=false, bool

Aufbruchsverknuepfung=false, bool Linienendverknuepfung=false, bool Basislinienwahl=false)

- void **holenLinienplanModulParameter** (int aufruf, int& gesamtlaenge, int& maxLinienLaenge, int& maxLinienAnzahl, double& linienplanID, int& geschwindigkeit, int& algorithmus, int& anzeigegraphen, bool& EindeutigeLinienverkn, bool& MehrdeutigeLinienverkn, bool& Aufbruchsverknuepfung, bool& Linienendverknuepfung, bool& Basislinienwahl)
- void **speichernPflichtlinienDatenbank** (list<**CLinie**∗>∗ pflichtlinien)
- void **speichernPflichtlinienGRASS** (list<**CLinie**∗>∗ pflichtlinien)
- void **speichernEinePflichtlinie** (list<int>∗ pflichtlinie, int id)
- list<**CLinie**∗>∗ **holenPflichtlinien** ()
- four_tuple<int,int,int,int>∗ **fahrplanIntervall** ()
- h_array<int,list<three_tuple<int,int,int>∗ >∗>∗ **fahrplanMindestumsteigezeit** ()
- h_array<int,int>∗ **fahrplanMindesthaltezeit** ()
- void **speichernFahrplanMindestumsteigezeit** (h_array<int,list<three_tuple<int,int,int>∗ >∗>∗)
- void **speichernFahrplanMindesthaltezeit** (h_array<int,int>∗)
- array<three_tuple<int,int,int>∗ >∗ **fahrplanAbfahrtszeiten** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- list<four_tuple<int,int,int,int>∗ >∗ **fahrplanDirekterAnschluss** ()
- void **holenAusgangsloesung** (double& summeNetzbedingteWartezeiten, double& wartezeitAufwand, double& mindestWartezeitAufwand, double& gesamtWarteaufwand)
- void **holenAnzahlFahrzeuge** (int& anzahlFahrzeuge)
- list<**CFahrplanElement**∗>∗ **holenFahrplanHaltestellen** ()
- list<**CFahrplanElement**∗>∗ **holenFahrplanLinien** ()
- void **speichernFahrplanModulParameter** (int maxfahrzeuganzahl, int allgemeinertakt, int mindesthaltezeit, int mindestumsteigezeit)
- void **holenFahrplanModulParameter** (int& maxfahrzeuganzahl, int& allgemeinertakt, int& mindesthaltezeit, int& mindestumsteigezeit)
- int **holenKnotenID** (double, double)
- **CPunkt**∗ **holenKnotenKoordinaten** (int)
- list<int>∗ **holenLinienID** (int)
- list<int>∗ **holenMoeglicheZielHaltestellen** (int)
- int **holenZusammenhangsKomponentenID** (int,int)
- list<int>∗ **holenEingehendeLinien** (int)
- list<int>∗ **holenAusgehendeLinien** (int)
- list<int>∗ **holenLinienHaltestellen** (int linienid)
- void **speichernAusschnittStrassennetz** ()
- void **loeschenLinie** (int id)
- **CReisebewertung**∗ **routenplanung** (int starths, int endhs)
- void **setzenAktuelleProjektID** (int id)
- int **holenAktuelleProjektID** ()
- void **setztenPhase** (int phase)
- int **holenPhase** ()

- int **anlegenNeuesProjekt** (int id, char* beschreibung, int phase)
- void **holenProjektInformationen** (int id, char*& beschreibung, int& phase)
- list<two_tuple<int,char*>*>* **holenAlleProjektInformationen** ()
- int **holenAnzahlProjekte** ()
- int **holenErsteFreieProjektID** ()
- void **loeschenProjekt** (int id)
- int **kopierenProjekt** (int quellid, int zielid, char* beschreibung=0)
- int **anlegenNeuesProjektPG369** (int id, char* beschreibung, int phase)
- void **holenProjektInformationenPG369** (int id, char*& beschreibung, int& phase)
- list<two_tuple<int,char*>*>* **holenAlleProjektInformationenPG369** ()
- void **loeschenProjektPG369** (int id)
- int **holenErsteFreieProjektIDPG369** ()
- int **kopierenProjektVonPG369** (int quellid, int zielid, char* beschreibung=0)
- int **kopierenProjektNachPG369** (int quellid, int zielid, char* beschreibung=0)
- void **speichernPflichthaltestellen** ()
- void **holenPflichthaltestellen** ()
- void **speichernAusschnitt** (double links, double oben, double rechts, double unten)
- void **holenAusschnitt** (double& links, double& oben, double& rechts, double& unten)
- void **aktuallisierenAnzeigen** (int phase)
- void **guiErgebnisrueckgabe** ()

4.25.1 Ausführliche Beschreibung

Diese Schnittstelle ermoeglicht der GUI den Zugriff auf die Datenbank und die Kontrolle (z.B: Stop von Threads) .

4.25.2 Beschreibung der Konstruktoren und Destruktoren

4.25.2.1 CGUISchnittstelle::CGUISchnittstelle (CDatenbank * *datenbank*)

4.25.3 Dokumentation der Elementfunktionen

4.25.3.1 void CGUISchnittstelle::aktuallisierenAnzeigen (int *phase*)

4.25.3.2 int CGUISchnittstelle::anlegenNeuesProjekt (int *id*, char * *beschreibung* = 0, int *phase* = 0)

4.25.3.3 int CGUISchnittstelle::anlegenNeuesProjektPG369 (int *id*, char * *beschreibung*, int *phase*)

4.25.3.4 array< three_tuple< int,int,int >* >* CGUISchnittstelle::fahrplanAbfahrtszeiten ()

4.25.3.5 list< four_tuple< int,int,int,int >* >* CGUISchnittstelle::fahrplanDirekterAnschluss ()

4.25.3.6 four_tuple< int,int,int,int >* CGUISchnittstelle::fahrplanIntervall ()

4.25.3.7 int CGUISchnittstelle::fahrplanMaximaleFahrzeuganzahl ()

4.25.3.8 h_array< int,int >* CGUISchnittstelle::fahrplanMindesthaltezeit ()

4.25.3.9 h_array< int,list< three_tuple< int,int,int >* >*>* CGUISchnittstelle::fahrplanMindestumsteigezeit ()

4.25.3.10 void CGUISchnittstelle::guiErgebnisrueckgabe ()

Diese Funktion wird vom verbundenen GUIObjekt aufgerufen, das seine berechneten Werte in der entsprechenden Datenstruktur zurueckgibt. Diese Funktion leitet die ihr uebergebenen Werte an die entsprechenden Stellen zur Speicherung bzw. zur Anzeige weiter

4.25.3.11 int CGUISchnittstelle::holenAktuelleProjektID ()

4.25.3.12 list< two_tuple< int,char *>*>* CGUISchnittstelle::holenAlleProjektInformationen ()

4.25.3.13 list< two_tuple< int,char *>*>* CGUISchnittstelle::holenAlleProjektInformationenPG369 ()

4.25.3.14 void CGUISchnittstelle::holenAnzahlFahrzeuge (int & *anzahlFahrzeuge*)

4.25.3.15 int CGUISchnittstelle::holenAnzahlProjekte ()

4.25.3.16 void CGUISchnittstelle::holenAusgangsloesung (double & *summeNetzbedingteWartezeiten*, double & *wartezeitAufwand*, double & *mindestWartezeitAufwand*, double & *gesamtWarteaufwand*)

4.25.3.17 list< int >* CGUISchnittstelle::holenAusgehendeLinien (int *id*)

4.25.3.18 void CGUISchnittstelle::holenAusschnitt (double & *links*, double & *oben*, double & *rechts*, double & *unten*)

4.25.3.19 list< int >* CGUISchnittstelle::holenEingehendeLinien (int *id*)

4.25.3.20 int CGUISchnittstelle::holenErsteFreieProjektID ()

4.25.3.21 int CGUISchnittstelle::holenErsteFreieProjektIDPG369 ()

4.25.3.22 list< CFahrplanElement *>* CGUISchnittstelle::holenFahrplanHaltestellen ()

4.25.3.23 list< CFahrplanElement *>* CGUISchnittstelle::holenFahrplanLinien ()

4.25.3.24 void CGUISchnittstelle::holenFahrplanModulParameter (int & *maxfahrzeuganzahl*, int & *allgemeinertakt*, int & *mindesthaltezeit*, int & *mindestumsteigezeit*)

4.25.3.25 CGuetemasse * CGUISchnittstelle::holenGuetemasse (void)

4.25.3.26 void CGUISchnittstelle::holenHaltestellenModulParameter (int *aufruf*, int & *algorithmus*, int & *zufallsgeneratorInitialisierung*, double & *minRadius*, double & *maxRadius*, double & *verbindungsRadiusStadt*, double & *verbindungsRadiusLand*, int & *radiustyp*, double & *haltestellenLage*, double & *toleranzgrenze*, int & *bevoelkerungsRaster*, double & *koeffizientBevoelkerungsueberversorgung*, double & *koeffizientBevoelkerungsunterversorgung*, double & *koeffizientFlaechenueberversorgung*, double & *koeffizientFlaechenunterversorgung*, double & *koeffizientFahrtwunscheinbeziehung*, double & *koeffizientStrassenanpassung*,

int & *AnzahlHaltestellen*, bool & *AnzStrassenkarte*, bool & *AnzBevoelkerungsversorgung*, bool & *AnzFlaechenversorgung*, bool & *Anzkomplettunterdruecken*, int & *Strategie*, int & *AnzahlEltern*, int & *AnzahlKinder*, int & *Generationen*, bool & *AbbruchGueteunterschreitung*, double & *GrenzwGueteunterschreitung*)

4.25.3.27 int CGUISchnittstelle::holenKnotenID (double *x*, double *y*)

4.25.3.28 CPunkt * CGUISchnittstelle::holenKnotenKoordinaten (int *id*)

4.25.3.29 list< int >* CGUISchnittstelle::holenLinienHaltestellen (int *linienid*)

4.25.3.30 list< int >* CGUISchnittstelle::holenLinienID (int *id*)

4.25.3.31 void CGUISchnittstelle::holenLinienplanModulParameter (int *aufruf*, int & *gesamtlaenge*, int & *maxLinienLaenge*, int & *maxLinienAnzahl*, double & *linienplanID*, int & *geschwindigkeit*, int & *algorithmus*, int & *anzeigegraphen*, bool & *EindeutigeLinienverkn*, bool & *MehrdeutigeLinienverkn*, bool & *Aufbruchsverknuepfung*, bool & *Linienendverknuepfung*, bool & *Basislinienwahl*)

4.25.3.32 list< int >* CGUISchnittstelle::holenMoeglicheZielHaltestellen (int *id*)

4.25.3.33 void CGUISchnittstelle::holenPflichthaltestellen (void)

4.25.3.34 list< CLinie *>* CGUISchnittstelle::holenPflichtlinien ()

4.25.3.35 int CGUISchnittstelle::holenPhase ()

4.25.3.36 void CGUISchnittstelle::holenProjektInformationen (int *id*, char *& *beschreibung*, int & *phase*)

4.25.3.37 void CGUISchnittstelle::holenProjektInformationenPG369 (int *id*, char *& *beschreibung*, int & *phase*)

4.25.3.38 int CGUISchnittstelle::holenZusammenhangs-KomponentenID (int *h*, int *l1*)

4.25.3.39 int CGUISchnittstelle::kopierenProjekt (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.25.3.40 int CGUISchnittstelle::kopierenProjektNachPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.25.3.41 int CGUISchnittstelle::kopierenProjektVonPG369 (int *quellid*, int *zielid*, char * *beschreibung* = 0)

4.25.3.42 void CGUISchnittstelle::loeschenLinie (int *id*)

4.25.3.43 void CGUISchnittstelle::loeschenProjekt (int *id*)

4.25.3.44 void CGUISchnittstelle::loeschenProjektPG369 (int *id*)

4.25.3.45 CReisebewertung * CGUISchnittstelle::routenplanung (int *starths*, int *endhs*)

4.25.3.46 void CGUISchnittstelle::setzenAktuelleProjektID (int *id*)

4.25.3.47 void CGUISchnittstelle::setztenPhase (int *phase*)

4.25.3.48 void CGUISchnittstelle::speichernAusschnitt (double *links*, double *oben*, double *rechts*, double *unten*)

4.25.3.49 void CGUISchnittstelle::speichernAusschnittStrassennetz ()

4.25.3.50 void CGUISchnittstelle::speichernBevoelkerungsDichte (list< CBevoelkerungsdichteElement *>* *bd*)

4.25.3.51 void CGUISchnittstelle::speichernEinePflichtlinie (list< int >* *pflichtlinie*, int *id*)

4.25.3.52 void CGUISchnittstelle::speichernFahrplan-Mindesthaltezeit (h_array< int,int >* *mh*)

4.25.3.53 void CGUISchnittstelle::speichernFahrplan-Mindestumsteigezeit (h_array< int,list< three_tuple< int,int,int >* >*>* *mu*)

4.25.3.54 void CGUISchnittstelle::speichernFahrplanModul-Parameter (int *maxfahrzeuganzahl*, int *allgemeinertakt*, int *mindest-haltezeit*, int *mindestumsteigezeit*)

4.25.3.55 void CGUISchnittstelle::speichernGuetemasse (CGuete-masse * *guetemasse*)

4.25.3.56 void CGUISchnittstelle::speichernHaltestellenModul-Parameter (int *aufruf*, int *algorithmus*, int *zufallsgenerator-Initialisierung*, double *minRadius*, double *maxRadius*, double *verbindungsRadiusStadt*, double *verbindungsRadiusLand*, int *radius-typ*, double *haltestellenLage*, double *toleranzgrenze*, int *bevoelkerungs-Raster*, double *koeffizientBevoelkerungsueberversorgung*, double *koeffizientBevoelkerungsunterversorgung*, double *koeffizientFlaechenueberversorgung*, double *koeffizientFlaechenunterversorgung*, double *koeffizient-Fahrtwunscheinbeziehung*, double *koeffizientStrassenanpassung*, int *AnzahlHaltestellen*, bool *AnzStrassenkarte*, bool *Anz-Bevoelkerungsversorgung*, bool *AnzFlaechenversorgung*, bool *Anzkomplettunterdruecken*, int *Strategie*, int *AnzahlEltern*, int *AnzahlKinder*, int *Generationen*, bool *AbbruchGueteunterschreitung*, double *GrenzwGueteunterschreitung*)

4.25.3.57 void CGUISchnittstelle::speichernLinienplanModul-Parameter (int *aufruf*, int *gesamtlaenge*, int *maxLinienLaenge*, int *maxLinienAnzahl*, double *linienplanID*, int *geschwindigkeit* = 15, int *algorithmus* = 0, int *anzeigegraphen* = 0, bool *EindeutigeLinienverkn* = false, bool *MehrdeutigeLinienverkn* = false, bool *Aufbruchs-verknuepfung* = false, bool *Linienendverknuepfung* = false, bool *Basislinienwahl* = false)

4.25.3.58 void CGUISchnittstelle::speichernPendlermatrix (matrix * *pendlerMatrix*, list< CPendlerListeElement *>* *pendlerListe*)

4.25.3.59 void CGUISchnittstelle::speichernPflichthaltestellen ()

4.25.3.60 void CGUISchnittstelle::speichernPflichtlinienDatenbank (list< CLinie *>* *pflichtlinien*)

4.25.3.61 void CGUISchnittstelle::speichernPflichtlinienGRASS (list< CLinie *>* *pflichtlinien*)

4.25.3.62 void CGUISchnittstelle::speichernUrsprungsQZMatrix (matrix * *quelleZielMatrix*, list< CQuelleZielListeElement *>* *quelleZielListe*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CGUISchnittstelle.h**
- **CGUISchnittstelle.cpp**

4.26 CHaltestelle Klassenreferenz

`#include <CHaltestelle.h>`

Öffentliche Datenelemente

- void **starten** (void)
- void **stoppen** (void)
- **CHaltestelle** (**CHSSchnittstelle** * lnkSchnittstelle)
- **~CHaltestelle** (void)

4.26.1 Ausführliche Beschreibung

Schnittstelle zur Programmgruppe. Hat die Kontrolle über das Package Haltestelle und alle darin enthaltenen Klasse.

4.26.2 Beschreibung der Konstruktoren und Destruktoren

4.26.2.1 CHaltestelle::CHaltestelle (CHSSchnittstelle * *lnkSchnittstelle*)

Konstruktor. Holt sich die Daten aus der Schnittstelle (mit CHSSchnittstelle.anfordernWerte()). Baut eigene Datenstrukturen (Fahrtwunschmatrix, Pendlermatrix, Strassennetz und Wegenetzgraph) auf. Setzt die Pflichthaltestellen.

Parameter:

lnkSchnittstelle Die Schnittstelle, welche von aussen uebergeben wird. Die Vernichtung erfolgt ebenfalls ausserhalb.

4.26.2.2 CHaltestelle::~CHaltestelle (void)

Destruktor. Dient dem Aufraeumen der Datenstrukturen.

4.26.3 Dokumentation der Elementfunktionen

4.26.3.1 void CHaltestelle::starten (void)

Leitet die Berechnung der Haltestellen und darauf aufbauender Algorithmen ein.

- ruft berechnenKreuzungen(), berechnenEntfernungen(), setzen-Haltestellen() und ermittelnKuerzesteWege() auf.
- kontrolliert die Berechnungen durch die Methodenaufrufe
- beruecksichtigt das Stoppen durch die Methode **stoppen**() (S. 96)

4.26.3.2 void CHaltestelle::stoppen (void)

Signalisiert das Stoppen der Haltestellenberechnung und darauf aufbauender Algorithmen.

Setzt dann ein Flag, das von der **starten**() (S. 95)-Methode ausgewertet wird

Wird vom Programm aufgerufen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CHaltestelle.h**
- **CHaltestelle.cpp**

4.27 CHaltestelleExceptions Klassenreferenz

`#include <Fehler.h>`

Öffentliche Datenelemente

- **CHaltestelleExceptions** ()
- **CHaltestelleExceptions** (string)
- string **gebenMeldung** ()

4.27.1 Ausführliche Beschreibung

Default Exception der Haltestellegruppe, es wird im Konstruktor eine Meldung uebergeben

4.27.2 Beschreibung der Konstruktoren und Destruktoren

4.27.2.1 CHaltestelleExceptions::CHaltestelleExceptions () `[inline]`

4.27.2.2 CHaltestelleExceptions::CHaltestelleExceptions (string *meldung*)

Der Konstruktor enthaelt eine Meldung zur Fehler beschreibung

4.27.3 Dokumentation der Elementfunktionen

4.27.3.1 string CHaltestelleExceptions::gebenMeldung ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.28 CHeuristischesLoesungsverfahren Klassenreferenz

`#include <CHeuristischesLoesungsverfahren.h>`

Klassendiagramm für CHeuristischesLoesungsverfahren:

Öffentliche Datenelemente

- **CHeuristischesLoesungsverfahren** (int, **CTransportkettengraph***)
- **~CHeuristischesLoesungsverfahren** ()
- void **starten** ()
- void **stoppen** ()

4.28.1 Ausführliche Beschreibung

Diese Klasse beschreibt die Fahrplan Erstellung per Heuristik

4.28.2 Beschreibung der Konstruktoren und Destruktoren

4.28.2.1 CHeuristischesLoesungsverfahren::CHeuristisches-Loesungsverfahren (int *zusammenhangskomponentenID*, CTransportkettengraph * *transportkettengraph*)

Konstruktor der Klasse

4.28.2.2 CHeuristischesLoesungsverfahren::~CHeuristisches-Loesungsverfahren ()

Destruktor der Klasse

4.28.3 Dokumentation der Elementfunktionen

4.28.3.1 void CHeuristischesLoesungsverfahren::starten () [virtual]

Wir rufen die Methode berechnenMaximalgeruest in **CTransportkettengraph** (S. 200) auf. Diese Methode erwartet als Parameter die ID der Zusammenhangskomponente. Im Anschluss daran rufen wir die Methode berechnenSimplex-Tableau auf in CTranspoetkettengraph auf. Diese erwartet als Eingabe die ID der Zusammenhangskomponente.

Implementiert von **CLoesungsgenerator** (S. 117).

4.28.3.2 void CHeuristischesLoesungsverfahren::stoppen () [virtual]

Mit dieser Methode wird die Aktuelle Berechnung beendet.

!!!!!!!!!!!!!!!!!!!!!!!!!!!! MUSS NOCH UEBERARBEITET WERDEN !!!!!!!!!!!!!!!!!!!!!!!!!!!!

Implementiert von **CLoesungsgenerator** (S. 117).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CHeuristischesLoesungsverfahren.h**
- **CHeuristischesLoesungsverfahren.cpp**

4.29 CHSSchnittstelle Klassenreferenz

`#include <CHSSchnittstelle.h>`

Klassendiagramm für CHSSchnittstelle:

Öffentliche Datenelemente

- **CHSSchnittstelle** (**CDatenbank**∗, **CGUI**∗)
- **CParameterHaltestelle**∗ **anfordernWerte** (void)
- void **haltestellenErgebnisrueckgabe** (GRAPH<**CWegenetz-Knoten**∗, **CWegenetzKante**∗>∗, matrix∗)
- void **speichernGuetemasse** (**CGuetemasse** ∗)
- **CGuetemasse**∗ **holenGuetemasse** ()

4.29.1 Ausführliche Beschreibung

Diese Klasse bildet die Schnittstelle zwischen dem Haltestellenmodul, der GUI und der Datenbank. Sie leitet Anforderungen und Daten zwischen den anfragenden und den antwortenden Klassen weiter. Diese Klasse reagiert nur auf Anfrage des Haltestellenmoduls und arbeitet nicht selbststaendig.

4.29.2 Beschreibung der Konstruktoren und Destruktoren

4.29.2.1 CHSSchnittstelle::CHSSchnittstelle (CDatenbank * *datenbank*, CGUI * *gui*)

4.29.3 Dokumentation der Elementfunktionen

4.29.3.1 CParameterHaltestelle * CHSSchnittstelle::anfordernWerte (void)

4.29.3.2 void CHSSchnittstelle::haltestellenErgebnisrueckgabe (GRAPH< CWegenetzKnoten *,CWegenetzKante *>*, matrix *)

Diese Funktion wird vom verbundenen Haltestellenmodul aufgerufen, das seine berechneten Werte in der entsprechenden Datenstruktur zurueckgibt. Diese Funktion leitet die ihr uebergebenen Werte an die entsprechenden Stellen zur Speicherung bzw. zur Anzeige weiter

Der zweite Parameter entspricht der Kantenbeschriftung, was hier der Entfernung entspricht

4.29.3.3 CGuetemasse * CHSSchnittstelle::holenGuetemasse (void)

4.29.3.4 void CHSSchnittstelle::speichernGuetemasse (CGuetemasse * *guetemasse*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CHSSchnittstelle.h**
- **CHSSchnittstelle.cpp**

4.30 CImport Klassenreferenz

`#include <CImport.h>`

Öffentliche Datenelemente

- void **importierenQuelleZielMatrix** ()

4.30.1 Ausführliche Beschreibung

importiert Daten aus Textdateien in die Datenbank

4.30.2 Dokumentation der Elementfunktionen

4.30.2.1 void CImport::importierenQuelleZielMatrix ()

Die folgende Funktion importiert die ihr uebergebenen Quelle-Ziel-Matrix Datei.
Der Rueckgabewert ist dabei die vollstaendige Quelle-Ziel-Matrix

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CImport.h**
- **CImport.cpp**

4.31 CKantenBeschriftung Klassenreferenz

`#include <CKantenBeschriftung.h>`

Öffentliche Datenelemente

- **CKantenBeschriftung** ()
- **CKantenBeschriftung** (int, int=0, bool=false)
- **CKantenBeschriftung** (const CKantenBeschriftung&)
- CKantenBeschriftung& **operator=** (const CKantenBeschriftung&)
- void **setzenIstUmsteigekante** ()
- bool **gebenIstUmsteigekante** ()
- void **setzenVerkehrsstrom** (int)
- void **setzenFesteVerbindung** (bool)
- bool **abfragenFesteVerbindung** ()
- int **abfragenVerkehrsstrom** ()
- bool **gebenImMaximalgeruest** ()
- void **setzenImMaximalgeruest** (bool)
- void **setzenKantenID** (int)
- int **gebenKantenID** ()
- void **setzenPositionImSimplexTableau** (int)
- int **gebenPositionImSimplexTableau** ()
- int **gebenNetzbedingteWartezeit** ()
- void **setzenNetzbedingteWartezeit** (int)
- void **setzenMindestHaltezeit** (int)
- void **setzenMindestUmsteigezeit** (int)
- int **gebenMindestHaltezeit** ()
- int **gebenMindestUmsteigezeit** ()
- int **gebenWartezeit** ()
- bool **istNichtImMaximalgeruest** ()
- void **setzenNichtImMaximalgeruest** (bool)
- bool **gebenMussInsMaximalgeruest** ()
- void **setzenMussInsMaximalgeruest** (bool)

Freundbeziehungen

- istream& **operator>>** (istream&, CKantenBeschriftung&)
- ostream& **operator<<** (ostream&, const CKantenBeschriftung&)

4.31.1 Ausführliche Beschreibung

Diese Kante dient der Speicherung von Kanten - Informationen des Transportkettengraphs

4.31.2 Beschreibung der Konstruktoren und Destruktoren

4.31.2.1 CKantenBeschriftung::CKantenBeschriftung () [inline]

4.31.2.2 CKantenBeschriftung::CKantenBeschriftung (int *kantenID*, int *verkehrsstrom* = 0, bool *istUmsteigeKante* = false)

Konstruktor

4.31.2.3 CKantenBeschriftung::CKantenBeschriftung (const CKantenBeschriftung & *kantenBeschriftungOriginal*)

Kopier-Konstruktor

4.31.3 Dokumentation der Elementfunktionen

4.31.3.1 bool CKantenBeschriftung::abfragenFesteVerbindung ()

GEBEN-Methode des festeVerbindung Attributes

4.31.3.2 int CKantenBeschriftung::abfragenVerkehrsstrom ()

GEBEN-Methode des verkehrsstrom Attributes

4.31.3.3 bool CKantenBeschriftung::gebenImMaximalgeruest ()

Geben-Methode des Attributes imMaximalgeruest

4.31.3.4 bool CKantenBeschriftung::gebenIstUmsteigekante ()

GEBEN-Methode istUmsteigekante

4.31.3.5 int CKantenBeschriftung::gebenKantenID ()

Mittels dieser Methode geben wir die ID der Kante heraus.

4.31.3.6 int CKantenBeschriftung::gebenMindestHaltezeit ()

Diese Methode gibt den Wert des Attributs mindestHaltezeit zurueck

4.31.3.7 int CKantenBeschriftung::gebenMindestUmsteigezeit ()

Diese Methode gibt den Wert des Attributs mindestUmsteigezeit zurueck

4.31.3.8 bool CKantenBeschriftung::gebenMussInsMaximalgeruest ()

Geben-Methode des Attributes mussInsMaximalgeruest

4.31.3.9 int CKantenBeschriftung::gebenNetzbedingteWartezeit ()

Gibt die netzbedingte Wartezeit der Kante zurueck

4.31.3.10 int CKantenBeschriftung::gebenPositionImSimplex-Tableau ()

Gibt den Inhalt der Variablen positionImSimplextableau zurueck.

4.31.3.11 int CKantenBeschriftung::gebenWartezeit ()

Diese Methode gibt die gesamte Wartezeit einer Kante zurueck Umsteigezeit + Haltezeit + NetzbedingteWartezeit

4.31.3.12 bool CKantenBeschriftung::istNichtImMaximalgeruest ()

Geben-Methode des Attributes nichtImMaximalgeruest

4.31.3.13 CKantenBeschriftung & CKanten-Beschriftung::operator= (const CKantenBeschriftung & *kanten-BeschriftungOriginal*)

= Operator

4.31.3.14 void CKantenBeschriftung::setzenFesteVerbindung (bool *setzen*)

SETZEN-Methode des festeVerbindung - Attributes

4.31.3.15 void CKantenBeschriftung::setzenImMaximalgeruest (bool *setzen*)

Setzen-Methode des Attributes imMaximalgeruest

4.31.3.16 void CKantenBeschriftung::setzenIstUmsteigekante ()

Setzt das Flag istUmsteigekante auf true

4.31.3.17 void CKantenBeschriftung::setzenKantenID (int *id*)

Mittels dieser Methode setzen wir die ID der Kante.

4.31.3.18 void CKantenBeschriftung::setzenMindestHaltezeit (int *wartezeit*)

Diese Methode setzt das Attribut mindestHaltezeit auf den uebergebenen Wert

4.31.3.19 void CKantenBeschriftung::setzenMindestUmsteigezeit (int *wartezeit*)

Diese Methode setzt das Attribut mindestUmsteigezeit auf den uebergebenen Wert

4.31.3.20 void CKantenBeschriftung::setzenMussIns-Maximalgeruest (bool *eintragen*)

Setzen-Methode des Attributes mussInsMaximalgeruest

4.31.3.21 void CKantenBeschriftung::setzenNetzbedingteWartezeit (int *wartezeit*)

Diese Methode setzt das Attribut netzbedingteWartezeit auf den uebergebenen Wert

4.31.3.22 void CKantenBeschriftung::setzenNichtIm-Maximalgeruest (bool *austragen*)

Setzen-Methode des Attributes nichtImMaximalgeruest

4.31.3.23 void CKantenBeschriftung::setzenPositionImSimplex-Tableau (int *position*)

Setzt die Variable positionImSimplextableau. Wird auf jeden Fall von einfuegen-KantenInSimplextableau benutzt.

4.31.3.24 void CKantenBeschriftung::setzenVerkehrsstrom (int *verkehrsstrom*)

SETZEN-Methode des verkehrsstrom - Attributes

4.31.4 Freundbeziehungen und Funktionsdokumentation

4.31.4.1 ostream & operator<< (ostream & *outstream*, const CKantenBeschriftung & *beschriftung*) [friend]

<< Operator

4.31.4.2 istream & operator>> (istream & *instream*, CKantenBeschriftung & *beschriftung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CKantenBeschriftung.h**
- **CKantenBeschriftung.cpp**

4.32 CKnoten Klassenreferenz

`#include <CKnoten.h>`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CKnoten.h**

4.33 CKnotenBeschriftung Klassenreferenz

`#include <CKnotenBeschriftung.h>`

Öffentliche Datenelemente

- void **setzenTempVerkehrsstrom** (int)
- int **gebenTempVerkehrsstrom** ()
- int **ausgebenHaltestellenID1** ()
- int **ausgebenHaltestellenID2** ()
- int **ausgebenLiniennummer** ()
- int **ausgebenFahrzeit** ()
- int **ausgebenAbfahrtzeit** ()
- void **setzenHaltestellenID1** (int)
- void **setzenHaltestellenID2** (int)
- void **setzenLiniennummer** (int)
- void **setzenFahrzeit** (int)
- void **setzenAbfahrtzeit** (int)
- void **setzenKnotenID** (int)
- int **gebenKnotenID** ()
- **CKnotenBeschriftung** (const CKnotenBeschriftung&)
- CKnotenBeschriftung& **operator=** (const CKnotenBeschriftung&)
- **CKnotenBeschriftung** ()

- **CKnotenBeschriftung** (int, int=0, int=0, int=0, int=0)
- void **initialisierenFahrplanzeitGesetzt** ()
- bool **ausgebenFahrplanzeitGesetzt** ()

Freundbeziehungen

- istream& **operator>>** (istream&, CKnotenBeschriftung&)
- ostream& **operator<<** (ostream&, const CKnotenBeschriftung&)

4.33.1 Ausführliche Beschreibung

Diese Klasse dient der Speicherung von Knoteninformationen des Transportkettengraphs

4.33.2 Beschreibung der Konstruktoren und Destruktoren

4.33.2.1 CKnotenBeschriftung::CKnotenBeschriftung (const CKnotenBeschriftung & *copy*)

Ueberladen des Copy-Konstruktors

4.33.2.2 CKnotenBeschriftung::CKnotenBeschriftung ()

4.33.2.3 CKnotenBeschriftung::CKnotenBeschriftung (int *knotenID*, int *liniennr* = 0, int *haltestellenID1* = 0, int *haltestellenID2* = 0, int *fahrzeit* = 0)

4.33.3 Dokumentation der Elementfunktionen

4.33.3.1 int CKnotenBeschriftung::ausgebenAbfahrtzeit ()

GEBEN-Methode des attributes Abfahrtzeit

4.33.3.2 bool CKnotenBeschriftung::ausgebenFahrplanzeitGesetzt ()

4.33.3.3 int CKnotenBeschriftung::ausgebenFahrzeit ()

GEBEN-Methode des fahrzeit Attributes

4.33.3.4 int CKnotenBeschriftung::ausgebenHaltestellenID1 ()

GEBEN-Methode des haltestellenID1 Attributes

4.33.3.5 int CKnotenBeschriftung::ausgebenHaltestellenID2 ()

GEBEN-Methode des haltestellenID2 Attributes

4.33.3.6 int CKnotenBeschriftung::ausgebenLiniennummer ()

GEBEN-Methode des liniennummer Attributes

4.33.3.7 int CKnotenBeschriftung::gebenKnotenID ()

GEBEN-Methode des Attributes KnotenID

4.33.3.8 int CKnotenBeschriftung::gebenTempVerkehrsstrom ()

GEBEN-Methode des tempVerkehrsstrom Attributes

4.33.3.9 void CKnotenBeschriftung::initialisierenFahrplanzeitGesetzt ()

4.33.3.10 CKnotenBeschriftung & CKnotenBeschriftung::operator= (const CKnotenBeschriftung & *copy*)

4.33.3.11 void CKnotenBeschriftung::setzenAbfahrtzeit (int *fahrzeit*)

SETZEN-Methode des Attributes Abfahrtszeit

4.33.3.12 void CKnotenBeschriftung::setzenFahrzeit (int *fahrzeit*)

SETZEN-Methode des fahrzeit Attributes

4.33.3.13 void CKnotenBeschriftung::setzenHaltestellenID1 (int *id*)

SETZEN-Methode des haltestellenID1 Attributes

4.33.3.14 void CKnotenBeschriftung::setzenHaltestellenID2 (int *id*)

SETZEN-Methode des haltestellenID2 Attributes

4.33.3.15 void CKnotenBeschriftung::setzenKnotenID (int *id*)

SETZEN-Methode des Attributes KnotenID

4.33.3.16 void CKnotenBeschriftung::setzenLiniennummer (int *nummer*)

SETZEN-Methode des liniennummer Attributes

4.33.3.17 void CKnotenBeschriftung::setzenTempVerkehrsstrom (int *verkehrsstrom*)

SETZEN-Methode des tempVerkehrsstrom Attributes

4.33.4 Freundbeziehungen und Funktionsdokumentation

4.33.4.1 ostream& operator<< (ostream & *os*, const CKnotenBeschriftung & *beschriftung*) [friend]

4.33.4.2 istream& operator>> (istream & *is*, CKnotenBeschriftung & *beschriftung*) [friend]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CKnotenBeschriftung.h**
- **CKnotenBeschriftung.cpp**

4.34 CKontrolle Klassenreferenz

`#include <CKontrolle.h>`

Öffentliche Datenelemente

- **CKontrolle** (**CGUISchnittstelle***&, **CGUI***, int, char*)
- **~CKontrolle** ()
- void **startenHaltestellenModul** ()
- void **startenLinienplanModul** ()
- void **startenFahrplanModul** ()
- void **startenFahrplanBerechnenHeuristischeFahrplanerstellung** (int zusammenhangskomponente)
- void **startenFahrplanAusgebenFahrplan** ()
- void **startenFahrplanSukzessiveLoesungsverbesserung** (int zusammenhangskomponente)
- void **startenFahrplanEntscheidungsbaumverfahren** (int zusammenhangskomponente, int entscheidungsbaumtiefe=10)
- void **startenFahrplanAusgebenBewertung** ()
- void **stoppenHaltestellenModul** ()
- void **zerstoerenHaltestellenModul** ()
- void **stoppenLinienplanModul** ()
- void **zerstoerenLinienplanModul** ()
- void **stoppenFahrplanModul** ()

- void **stoppenFahrplanBerechnenHeuristischeFahrplanerstellung** ()
- void **stoppenFahrplanAusgebenFahrplan** ()
- void **stoppenFahrplanSukzessiveLoesungsverbesserung** ()
- void **stoppenFahrplanEntscheidungsbaumverfahren** ()
- void **stoppenFahrplanAusgebenBewertung** ()
- void **zerstoerenFahrplanModul** ()
- void **ausgebenWartezeiten** ()
- void **ausgebenAnzahlFahrzeuge** ()
- void **setzenPhaseBatchModus** (int phase)

Geschützte Attribute

- **CDatenbank* lnkCDatenbank**
- **CGUI* lnkCGUI**
- int **isttestversion**
- **CGUISchnittstelle* lnkCGUISchnittstelle**
- struct **CKontrolle::haltestelleStruct lnkHaltestelle**
- struct **CKontrolle::linienplanStruct lnkLinienplan**
- struct **CKontrolle::fahrplanStruct lnkFahrplan**
- int **res**
- pthread_t **haltestellen_thread**
- pthread_t **linienplan_thread**
- pthread_t **fahrplan_thread**
- void* **thread_result**

Geschützte, statische Datenelemente

- void* **threadHaltestelle** (void*)
- void* **threadLinienplan** (void*)
- void* **threadFahrplan** (void*)
- void* **threadFahrplanBerechnenHeuristischeFahrplanerstellung** (void*)
- void* **threadFahrplanAusgebenFahrplan** (void*)
- void* **threadFahrplanSukzessiveLoesungsverbesserung** (void*)
- void* **threadFahrplanEntscheidungsbaumverfahren** (void*)
- void* **threadFahrplanAusgebenBewertung** (void*)

4.34.1 Ausführliche Beschreibung

verwaltet die Instanzen der einzelnen Module (Erschaffung, Init, Aufruf, Zerstoerung) verwaltet die Datenfluesse

4.34.2 Beschreibung der Konstruktoren und Destruktoren

4.34.2.1 CKontrolle::CKontrolle (CGUISchnittstelle *& *arg-GUISchnittstelle*, CGUI * *gui*, int *itv*, char * *db_benutzername* = 0)

Konstruktor leitet die Erzeugung der entsprechenden Schnittstellen ein, von denen ja immer nur ein Objekt zur Laufzeit exisitieren darf. Die GUI erzeugt ein CKontroll-Objekt, das seinerseits die Schnittstellen und die angeschlossenen Module verwaltet. Die GUI bekommt aber ebenfalls eine Schnittstelle, um auf die Datenbank zugreifen zu koennen. Die vom Kontrollobjekt erzeugte GUISchnittstelle wird dabei explizit durch den uebergebenen Zeiger an das erzeugende GUI-Objekt

GUI muss mit uebergeben werden, damit diese an die Schnittstellen weitergereicht werden kann

4.34.2.2 CKontrolle::~CKontrolle ()

Destruktor dient zum Aufraeumen, bevor das Programm terminiert

4.34.3 Dokumentation der Elementfunktionen

4.34.3.1 void CKontrolle::ausgebenAnzahlFahrzeuge ()

4.34.3.2 void CKontrolle::ausgebenWartezeiten ()

4.34.3.3 void CKontrolle::setzenPhaseBatchModus (int *phase*)

die Folgenden Funktionen dienen als Hilffunktionen fuer den Batchmodus

4.34.3.4 void CKontrolle::startenFahrplanAusgebenBewertung ()

4.34.3.5 void CKontrolle::startenFahrplanAusgebenFahrplan ()

4.34.3.6 void CKontrolle::startenFahrplanBerechnenHeuristische-Fahrplanerstellung (int *zusammenhangskomponente*)

4.34.3.7 void CKontrolle::startenFahrplan-Entscheidungsbaumverfahren (int *zusammenhangskomponente*, int *entscheidungsbaumtiefe* = 10)

4.34.3.8 void CKontrolle::startenFahrplanModul ()

4.34.3.9 void CKontrolle::startenFahrplanSukzessive-Loesungsverbesserung (int *zusammenhangskomponente*)

4.34.3.10 void CKontrolle::startenHaltestellenModul ()

die folgenden Funktionen dienen zum Starten des entsprechenden Moduls

4.34.3.11 void CKontrolle::startenLinienplanModul ()

4.34.3.12 void CKontrolle::stoppenFahrplanAusgebenBewertung ()

4.34.3.13 void CKontrolle::stoppenFahrplanAusgebenFahrplan ()

4.34.3.14 void CKontrolle::stoppenFahrplanBerechnenHeuristische-Fahrplanerstellung ()

4.34.3.15 void CKontrolle::stoppenFahrplan-Entscheidungsbaumverfahren ()

4.34.3.16 void CKontrolle::stoppenFahrplanModul ()

4.34.3.17 void CKontrolle::stoppenFahrplanSukzessive-Loesungsverbesserung ()

4.34.3.18 void CKontrolle::stoppenHaltestellenModul ()

die Folgenden Funktionen dienen zum Anhalten und anschliessendem zerstoeren der entsprechenden Module

4.34.3.19 void CKontrolle::stoppenLinienplanModul ()

4.34.3.20 void * CKontrolle::threadFahrplan (void * *arg*) [static, protected]

4.34.3.21 void * CKontrolle::threadFahrplanAusgebenBewertung (void * *arg*) [static, protected]

4.34.3.22 void * CKontrolle::threadFahrplanAusgebenFahrplan (void * *arg*) [static, protected]

4.34.3.23 void * CKontrolle::threadFahrplanBerechnenHeuristische-Fahrplanerstellung (void * *arg*) [static, protected]

4.34.3.24 void * CKontrolle::threadFahrplan-Entscheidungsbaumverfahren (void * *arg*) [static, protected]

4.34.3.25 void * CKontrolle::threadFahrplanSukzessive-Loesungsverbesserung (void * *arg*) [static, protected]

4.34.3.26 void * CKontrolle::threadHaltestelle (void * *arg*) [static, protected]

Diese folgenden drei Funktionen dienen als Hilffunktionen fuer die Threadausfuehrung. Das Vorgehen ist dabei Folgendermassen: Die start-Funktionen rufen eine Betriebssystemfunktion zum starten eines Threads auf. DIese BS-Funktion erwatet als Uebergabeparamter eine Funktion, die im Folgenden als eigenstaendiger Thread ausgefuehrt wird. Hier uebernehmen die drei Folgenden Funktionen diese Aufgabe. Innerhalb dieser Funktionen werden die Objekte mit ihren jeweiligen Funktionen angelegt und ausgefuehrt.

Als Argumente werden die Schnittstellen uebergeben, die dann an die einzelnen Module weiterreichen

4.34.3.27 void * CKontrolle::threadLinienplan (void * *arg*) [static, protected]

4.34.3.28 void CKontrolle::zerstoerenFahrplanModul ()

4.34.3.29 void CKontrolle::zerstoerenHaltestellenModul ()

4.34.3.30 void CKontrolle::zerstoerenLinienplanModul ()

4.34.4 Dokumentation der Datenelemente

4.34.4.1 pthread_t CKontrolle::fahrplan_thread [protected]

4.34.4.2 pthread_t CKontrolle::haltestellen_thread [protected]

4.34.4.3 int CKontrolle::isttestversion [protected]

4.34.4.4 pthread_t CKontrolle::linienplan_thread [protected]

4.34.4.5 CDatenbank * CKontrolle::lnkCDatenbank [protected]

4.34.4.6 CGUI * CKontrolle::lnkCGUI [protected]

4.34.4.7 CGUISchnittstelle * CKontrolle::lnkCGUISchnittstelle [protected]

4.34.4.8 struct CKontrolle::fahrplanStruct CKontrolle::lnkFahrplan [protected]

4.34.4.9 struct CKontrolle::haltestelleStruct CKontrolle::lnkHaltestelle [protected]

Schnittstelle und Modul werden in einer Struktur gespeichert, damit es a) uebersichtlicher wird und b) damit der Datenaustausch ueber die Threads klappt (die Thread-Funktion erlaubt nur ein Argument

4.34.4.10 struct CKontrolle::linienplanStruct CKontrolle::lnkLinienplan [protected]

4.34.4.11 int CKontrolle::res [protected]

4.34.4.12 void * CKontrolle::thread_result [protected]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CKontrolle.h**
- **CKontrolle.cpp**

4.35 CKreisGewaehlt Klassenreferenz

`#include <Fehler.h>`

Klassendiagramm für CKreisGewaehlt:

Öffentliche Datenelemente

- **CKreisGewaehlt** (string)

4.35.1 Ausführliche Beschreibung

Wird vom Fahrplanmodul benutzt, wenn der Benutzer bei der Auswahl von festen Bindungen dabei einen Kreis auswaehlt.

4.35.2 Beschreibung der Konstruktoren und Destruktoren

4.35.2.1 CKreisGewaehlt::CKreisGewaehlt (string *meldung*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.36 CLinie Klassenreferenz

`#include <CLinie.h>`

Öffentliche Datenelemente

- **CLinie** ()
- **CLinie** (const CLinie&)
- CLinie& **operator=** (const CLinie&)
- void **setzenID** (int id)
- int **holenID** ()
- void **setzenhListe** (list<int>* einehListe)
- list<int>* **holenhListe** ()
- int **holenLaenge** (**CNetzplan*** meinNetzplan)
- void **setzenPflicht** (bool)
- bool **holenPflicht** ()
- **LEDA_MEMORY** (CLinie)

Freundbeziehungen

- istream& **operator>>** (istream&, CLinie&)
- ostream& **operator<<** (ostream&, const CLinie&)

4.36.1 Beschreibung der Konstruktoren und Destruktoren

4.36.1.1 CLinie::CLinie ()

Konstruktor

4.36.1.2 CLinie::CLinie (const CLinie & *OriginalLinie*)

Kopier Konstruktor

4.36.2 Dokumentation der Elementfunktionen

4.36.2.1 CLinie::LEDA_MEMORY (CLinie)

4.36.2.2 int CLinie::holenID ()

4.36.2.3 int CLinie::holenLaenge (CNetzplan * *meinNetzplan*)

4.36.2.4 bool CLinie::holenPflicht ()

4.36.2.5 list< int >* CLinie::holenhListe ()

4.36.2.6 CLinie & CLinie::operator= (const CLinie & *OriginalLinie*)

= Operator

4.36.2.7 void CLinie::setzenID (int *id*)

4.36.2.8 void CLinie::setzenPflicht (bool *einPflicht*)

4.36.2.9 void CLinie::setzenhListe (list< int >* *einehListe*)

4.36.3 Freundbeziehungen und Funktionsdokumentation

4.36.3.1 ostream & operator<< (ostream & *os*, const CLinie & *AusgabeLinie*) [friend]

<< Operator

4.36.3.2 istream & operator>> (istream & *is*, CLinie & *ZielLinie*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CLinie.h**
- **CLinie.cpp**

4.37 CLinienplan Klassenreferenz

`#include <CLinienplan.h>`

Öffentliche Datenelemente

- **CLinienplan** (**CLPSchnittstelle*** programmSchnittstelle)
- **CLinienplan** (**CLPSchnittstelleTest*** programmSchnittstelle)

- ~**CLinienplan** ()
- void **starten** ()
- void **rueckkoppeln** ()
- void **stoppen** ()

4.37.1 Ausführliche Beschreibung

Dies ist die Hauptklasse des Linienplan - Moduls. Die komplette Kommunikation vom Programm - Modul zum Linienplan - Modul geschieht ueber diese Klasse. Desweiteren wird von dieser Klasse aus das Linienplan - Modul gesteuert.

4.37.2 Beschreibung der Konstruktoren und Destruktoren

4.37.2.1 CLinienplan::CLinienplan (CLPSchnittstelle * *programm-Schnittstelle*)

Der Konstruktor der Klasse CLinienplan. Er wird beim generieren der Klasse aufgerufen und sorgt fuer die Initialisierung der Klasse.

Eingabe:

- Ein Zeiger auf ein Objekt der Klasse **CLPSchnittstelle** (S. 118). Dieses Objekt stellt die Verbindung zum Modul Programm dar, an das auch spaeter das Ergebins geliefert werden soll.

Rueckgabe:

- nichts

4.37.2.2 CLinienplan::CLinienplan (CLPSchnittstelleTest * *programmSchnittstelle*)

Der Konstruktor fuer die Testumgebung

4.37.2.3 CLinienplan::~CLinienplan ()

Destruktor der Klasse CLinienplan. Das Objekt wird nicht mehr benoetigt und muss nun alle von ihm erzeugten Objekte loeschen.

Eingabe:

- nichts

Rueckgabe:

- nichts

4.37.3 Dokumentation der Elementfunktionen

4.37.3.1 void CLinienplan::rueckkoppeln ()

Dieses Linienplanobjekt hat schon einen Linienplan erzeugt, aber es soll ein neuer Linienplan unter Beruecksichtigung veraenderter Parameter berechnet werden. Dabei sind der Wegenetzgraph und die Verkehrsbedarfsmatrix von den Aenderungen ausgenommen.

Eingabe:

- nichts (werden vom Programm abgeholt)

Rueckgabe:

- nichts

4.37.3.2 void CLinienplan::starten (void)

Stoesst das Erzeugen eines Linienplans an.

Eingabe:

- nichts (werden vom Programm abgeholt)

Rueckgabe:

- nichts

4.37.3.3 void CLinienplan::stoppen (void)

Mit dieser Methode ist es moeglich zum naechst moeglichem Zeitpunkt die Berechnung des Linienplans abzubrechen. Alle bisher getaetigten Berechungen sind nicht nutzbar. Alle vom Linienplan erzeugten Objekte werden geloescht.

Eingabe:

- nichts

Rueckgabe:

- nichts

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CLinienplan.h**
- **CLinienplan.cpp**
- **linienplandummydaten.cpp**

4.38 CLoesungsgenerator Klassenreferenz

`#include <CLoesungsgenerator.h>`

Klassendiagramm für CLoesungsgenerator:

Öffentliche Datenelemente

- **CLoesungsgenerator** (int,**CTransportkettengraph**∗)
- virtual **~CLoesungsgenerator** ()
- virtual void **stoppen** () = 0
- virtual void **starten** () = 0

Geschützte Attribute

- **CTransportkettengraph**∗ **Transportkettengraph**
- int **zusammenhangskomponentenID**

4.38.1 Ausführliche Beschreibung

Bei dieser Klasse handelt es sich um die Grundstruktur der Berechnungsklassen. Diese wird mit den Klassen der einzelnen Verfahren initialisiert.

4.38.2 Beschreibung der Konstruktoren und Destruktoren

4.38.2.1 CLoesungsgenerator::CLoesungsgenerator (int *zusammenhangskomponentenID*, CTransportkettengraph ∗ *transportkettengraph*)

Konstruktor der Klasse Es wird die ID der Zusammenhangskomponente und der Transportkettengraph uebergeben

4.38.2.2 CLoesungsgenerator::~CLoesungsgenerator () [inline, virtual]

Destruktor der Klasse

4.38.3 Dokumentation der Elementfunktionen

4.38.3.1 virtual void CLoesungsgenerator::starten () [pure virtual]

Erneute Implementation in **CEntscheidungsbaumverfahren** (S. 54), **CHeuristischesLoesungsverfahren** (S. 97), und **CSukzessivVerfahren** (S. 178).

4.38.3.2 void CLoesungsgenerator::stoppen () [pure virtual]

Mit dieser Methode muss es möglich sein, die Berechnung zu beenden.

Erneute Implementation in **CEntscheidungsbaumverfahren** (S. 55), **CHeuristischesLoesungsverfahren** (S. 98), und **CSukzessivVerfahren** (S. 178).

4.38.4 Dokumentation der Datenelemente

4.38.4.1 CTransportkettengraph * CLoesungsgenerator::Transportkettengraph [protected]

Zeiger auf das Objekt der Klasse **CTransportkettengraph** (S. 200)

4.38.4.2 int CLoesungsgenerator::zusammenhangskomponentenID [protected]

Hier wird die Zusammenhangskomponenten ID gespeichert, fuer die eine Loesung berechnet werden soll.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CLoesungsgenerator.h**
- **CLoesungsgenerator.cpp**

4.39 CLPSchnittstelle Klassenreferenz

`#include <CLPSchnittstelle.h>`

Klassendiagramm für CLPSchnittstelle:

Öffentliche Datenelemente

- **CLPSchnittstelle** (**CDatenbank***, **CGUI***)
- matrix* **holenLinienplanQuelleZielMatrix** ()

- list<**CLinie**∗>∗ **holenLinienplanVorgegebeneLinien** ()
- UGRAPH<**CWegenetzKnoten**∗,**CWegenetzKante**∗>∗ **holenLinienplanWegenetzgraph** ()
- int **holenLinienplanGesamtlaenge** ()
- int **holenLinienplanLinienLaenge** ()
- int **holenLinienplanMaxLinienAnzahl** ()
- int **holenGeschwindigkeit** ()
- int **holenAlgorithmus** ()
- int **holenAnzeigenGraphen** ()
- bool **holenEindeutigeLinienverknuepfung** ()
- bool **holenMehrdeutigeLinienverknuepfung** ()
- bool **holenAufbruchsverknuepfung** ()
- bool **holenLinienendverknuepfung** ()
- bool **holenBasislinienwahl** ()
- bool **pruefenBatchModus** (void)
- void **linienplanungErgebnisrueckgabe** (GRAPH<**CFPlanGraphKnoten**∗, **CFPlanGraphKante**∗>∗, list<**CLinie**∗>∗)
- void **speichernGuetemasse** (**CGuetemasse** ∗)
- **CGuetemasse**∗ **holenGuetemasse** ()

4.39.1 Ausführliche Beschreibung

Diese Klasse bildet die Schnittstelle zwischen dem Linienplanmodul, der GUI und der Datenbank. Sie leitet Anforderungen und Daten zwischen den anfragenden und den antwortenden Klassen weiter. Diese Klasse reagiert nur auf Anfrage des Linienplanmoduls und arbeitet nicht selbststaendig.

4.39.2 Beschreibung der Konstruktoren und Destruktoren

4.39.2.1 CLPSchnittstelle::CLPSchnittstelle (CDatenbank ∗ *datenbank*, CGUI ∗ *gui*)

4.39.3 Dokumentation der Elementfunktionen

4.39.3.1 int CLPSchnittstelle::holenAlgorithmus ()

4.39.3.2 int CLPSchnittstelle::holenAnzeigenGraphen ()

4.39.3.3 bool CLPSchnittstelle::holenAufbruchsverknuepfung ()

4.39.3.4 bool CLPSchnittstelle::holenBasislinienwahl ()

4.39.3.5 bool CLPSchnittstelle::holenEindeutigeLinienverknuepfung ()

4.39.3.6 int CLPSchnittstelle::holenGeschwindigkeit ()

4.39.3.7 CGuetemasse * CLPSchnittstelle::holenGuetemasse (void)

4.39.3.8 bool CLPSchnittstelle::holenLinienendverknuepfung ()

4.39.3.9 int CLPSchnittstelle::holenLinienplanGesamtlaenge ()

Diese Funktionen werden vom verbundenen Linienplanmodul aufgerufen, um die entsprechenden ModulParameter von der Datenbank anzufordern

4.39.3.10 int CLPSchnittstelle::holenLinienplanLinienLaenge ()

4.39.3.11 int CLPSchnittstelle::holenLinienplanMaxLinienAnzahl ()

4.39.3.12 matrix * CLPSchnittstelle::holenLinienplanQuelleZiel-Matrix ()

Diese Funktionen werden vom verbundenen Linienplanmodul aufgerufen, um die entsprechenden Werte von der Datenbank anzufordern

4.39.3.13 list< CLinie *>* CLPSchnittstelle::holenLinienplan-VorgegebeneLinien ()

4.39.3.14 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CLPSchnittstelle::holenLinienplanWegenetzgraph ()

4.39.3.15 bool CLPSchnittstelle::holenMehrdeutige-Linienverknuepfung ()

4.39.3.16 void CLPSchnittstelle::linienplanungErgebnisrueckgabe (GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *fpgraph* = 0, list< CLinie *>* *buslinien* = 0)

Diese Funktion wird vom verbundenen Linienplanmodul aufgerufen, das seine berechneten Werte in der entsprechenden Datenstruktur zurueckgibt. Diese Funktion leitet die ihr uebergebenen Werte an die entsprechenden Stellen zur Speicherung bzw. zur Anzeige weiter

4.39.3.17 bool CLPSchnittstelle::pruefenBatchModus (void)

4.39.3.18 void CLPSchnittstelle::speichernGuetemasse (CGuetemasse * *guetemasse*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CLPSchnittstelle.h**
- **CLPSchnittstelle.cpp**

4.40 CLPSchnittstelleTest Klassenreferenz

`#include <CLPSchnittstelleTest.h>`

Öffentliche Datenelemente

- **CLPSchnittstelleTest** ()
- void **linienplanungErgebnisrueckgabe** (GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>*)

4.40.1 Beschreibung der Konstruktoren und Destruktoren

4.40.1.1 CLPSchnittstelleTest::CLPSchnittstelleTest ()

4.40.2 Dokumentation der Elementfunktionen

4.40.2.1 void CLPSchnittstelleTest::linienplanungErgebnisrueckgabe (GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *graph*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CLPSchnittstelleTest.h**
- **CLPSchnittstelleTest.cpp**

4.41 CNetzplan Klassenreferenz

`#include <CNetzplan.h>`

Öffentliche Datenelemente

- **CNetzplan** (bool batchmodus)
- **~CNetzplan** ()
- void **setzenMaxLinienlaenge** (int maxLaenge)
- int **holenMaxLinienlaenge** ()
- void **setzenMaxLinienanzahl** (int maxLinien)
- int **holenMaxLinienanzahl** ()
- void **setzenGesamtlinienLaenge** (int gesLaenge)

- int **holenGesamtlinienLaenge** ()
- void **setzenNetzplan** (UGRAPH<**CNetzplanKnoten***, **CNetzplanKante***>* netzgraph)
- UGRAPH<**CNetzplanKnoten***, **CNetzplanKante***>* **holenNetzplan** ()
- void **setzenLinienliste** (list<**CLinie***>* eineLinienliste)
- list<**CLinie***>* **holenLinienliste** ()
- void **setzenStop** (bool einstop)
- bool **holenStop** ()
- bool **pruefenZusammenhang** (node Startknoten)
- node_array<edge> **machenDijkstra** (node Start)
- void **vorbereitenDijkstra** ()
- node* **umwandelnHaltestelleZuKnoten** (UGRAPH<**CNetzplanKnoten***,**CNetzplanKante***>* wegenetzgraph, int haltestelle)
- edge* **suchenKanteZwischenKnoten** (UGRAPH<**CNetzplanKnoten***,**CNetzplanKante***>* wegenetzgraph, node* startKnoten, node* endKnoten)
- void **hinzufuegenLinie** (**CLinie*** linie)
- bool **testeLinienListe** ()
- void **pruefenFahrplangraph** (UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***> * wegenetzgraph, GRAPH<**CFPlanGraphKnoten***,**CFPlanGraphKante***>* fahrplangraph)
- int **testenAufZyklusfreiheit** (list<int> *linkeHListe,list<int> *rechteHListe)
- void **setzenGesamteStreckenkilometer** (int gesamteStreckenkm)
- int **holenGesamteStreckenkilometer** ()
- void **setzenMittlereMindestreisezeit** (int mittleremindestreisezt)
- int **holenMittlereMindestreisezeit** ()
- int **holenEntfernungsteiler** ()
- void **setzenGeschwindigkeit** (int geschwindigkeit)
- **CFortschritt*** **holenFortschritt** ()

4.41.1 Ausführliche Beschreibung

Modulglobale Datenstruktur Enthaelt Netzplan (LEDA::UGRAPH) und Linienliste (LEDA::list) Enthaelt die begrenzenden Parameter max. Linienlaenge, max Linienanz. und ges. Linienlaenge (uebergeben von **CLinienplan** (S. 114))

Aussehen des Netzplans:

- Knoten: ID, Prioritaet
- Kanten: Fahrzeit, Verkehrsstrom, Anzahl Basislinien

Aussehen der Linienliste: Liste von Clinie(n)

4.41.2 Beschreibung der Konstruktoren und Destruktoren

4.41.2.1 CNetzplan::CNetzplan (bool *batchmodus*)

4.41.2.2 CNetzplan::~CNetzplan ()

4.41.3 Dokumentation der Elementfunktionen

4.41.3.1 void CNetzplan::hinzufuegenLinie (CLinie * *linie*)

FIXME: Kommentar fehlt

4.41.3.2 int CNetzplan::holenEntfernungsteiler ()

4.41.3.3 CFortschritt * CNetzplan::holenFortschritt ()

4.41.3.4 int CNetzplan::holenGesamteStreckenkilometer ()

4.41.3.5 int CNetzplan::holenGesamtlinienLaenge ()

4.41.3.6 list< CLinie *>* CNetzplan::holenLinienliste ()

4.41.3.7 int CNetzplan::holenMaxLinienanzahl ()

4.41.3.8 int CNetzplan::holenMaxLinienlaenge ()

4.41.3.9 int CNetzplan::holenMittlereMindestreisezeit ()

4.41.3.10 UGRAPH< CNetzplanKnoten *,CNetzplanKante *>* CNetzplan::holenNetzplan ()

4.41.3.11 bool CNetzplan::holenStop ()

4.41.3.12 node_array< edge > CNetzplan::machenDijkstra (node *Start*)

FIXME: Kommentar fehlt

4.41.3.13 void CNetzplan::pruefenFahrplangraph (UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* *wegenetzgraph*, GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *fahrplangraph*)

FIXME: Kommentar fehlt

4.41.3.14 bool CNetzplan::pruefenZusammenhang (node *Startknoten*)

FIXME: Kommentar fehlt

4.41.3.15 void CNetzplan::setzenGesamteStreckenkilometer (int *gesamteStreckenkm*)

4.41.3.16 void CNetzplan::setzenGesamtlinienLaenge (int *gesLaenge*)

4.41.3.17 void CNetzplan::setzenGeschwindigkeit (int *geschwindigkeit*)

4.41.3.18 void CNetzplan::setzenLinienliste (list< CLinie *>* *eineLinienliste*)

4.41.3.19 void CNetzplan::setzenMaxLinienanzahl (int *maxLinien*)

4.41.3.20 void CNetzplan::setzenMaxLinienlaenge (int *maxlaenge*)

4.41.3.21 void CNetzplan::setzenMittlereMindestreisezeit (int *mittleremindestreisezt*)

4.41.3.22 void CNetzplan::setzenNetzplan (UGRAPH< CNetzplanKnoten *,CNetzplanKante *>* *netzgraph*)

4.41.3.23 void CNetzplan::setzenStop (bool *einstop*)

4.41.3.24 edge * CNetzplan::suchenKanteZwischenKnoten (UGRAPH< CNetzplanKnoten *,CNetzplanKante *>* *wegenetzgraph*, node * *startKnoten*, node * *endKnoten*)

FIXME: Kommentar fehlt

4.41.3.25 bool CNetzplan::testeLinienListe ()

FIXME: Kommentar fehlt

4.41.3.26 int CNetzplan::testenAufZyklusfreiheit (list< int >* *linkeHListe*, list< int >* *rechteHListe*)

FIXME: Kommentar fehlt

Rueckgabe: Anzahl der gleichen Haltestellen

4.41.3.27 node * CNetzplan::umwandelnHaltestelleZuKnoten (UGRAPH< CNetzplanKnoten *,CNetzplanKante *>* *wegenetz-graph*, int *haltestelle*)

FIXME: Kommentar fehlt

4.41.3.28 void CNetzplan::vorbereitenDijkstra ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CNetzplan.h**
- **CNetzplan.cpp**

4.42 CNetzplanKante Klassenreferenz

`#include <CNetzplanKante.h>`

Öffentliche Datenelemente

- **CNetzplanKante** ()
- **CNetzplanKante** (int eineEntfernung, int einVerkehrsstrom, int eine-AnzBasislinien, int eineId)
- **CNetzplanKante** (const CNetzplanKante&)
- CNetzplanKante& **operator=** (const CNetzplanKante&)
- void **setzenEntfernung** (int entf)
- void **setzenVerkehrsstrom** (int strom)
- void **setzenAnzBasislinien** (int anzahl)
- int **holenEntfernung** ()
- int **holenVerkehrsstrom** ()
- int **holenAnzBasislinien** ()
- void **setzenID** (int eineID)
- int **holenID** ()
- **LEDA_MEMORY** (CNetzplanKante)

Freundbeziehungen

- istream& **operator>>** (istream&, CNetzplanKante&)
- ostream& **operator<<** (ostream&, const CNetzplanKante&)
- int **compare** (const CNetzplanKante&, const CNetzplanKante&)

4.42.1 Ausführliche Beschreibung

Kantenobjekt des CNetzplans Enthaelt die Attribute Fahrzeit, Verkehrsstrom, Anzahl der Basislinien und jeweilige setzen- und geben-Operationen

4.42.2 Beschreibung der Konstruktoren und Destruktoren

4.42.2.1 CNetzplanKante::CNetzplanKante ()

Default Konstruktor

4.42.2.2 CNetzplanKante::CNetzplanKante (int *eineEntfernung*, int *einVerkehrsstrom*, int *eineAnzBasislinien*, int *eineId*)

Konstruktor mit initialisierung

4.42.2.3 CNetzplanKante::CNetzplanKante (const CNetzplanKante & *OriginalKante*)

Kopier Konstruktor

4.42.3 Dokumentation der Elementfunktionen

4.42.3.1 CNetzplanKante::LEDA_MEMORY (CNetzplanKante)

4.42.3.2 int CNetzplanKante::holenAnzBasislinien ()

4.42.3.3 int CNetzplanKante::holenEntfernung ()

4.42.3.4 int CNetzplanKante::holenID ()

4.42.3.5 int CNetzplanKante::holenVerkehrsstrom ()

4.42.3.6 CNetzplanKante & CNetzplanKante::operator= (const CNetzplanKante & *OriginalKante*)

= Operator

4.42.3.7 void CNetzplanKante::setzenAnzBasislinien (int *anzahl*)

4.42.3.8 void CNetzplanKante::setzenEntfernung (int *entf*)

4.42.3.9 void CNetzplanKante::setzenID (int *eineID*)

4.42.3.10 void CNetzplanKante::setzenVerkehrsstrom (int *strom*)

4.42.4 Freundbeziehungen und Funktionsdokumentation

4.42.4.1 int compare (const CNetzplanKante & *kante1*, const CNetzplanKante & *kante2*) [friend]

compare Operator

4.42.4.2 ostream & operator<< (ostream & *os*, const CNetzplanKante & *Ausgabekante*) [friend]

<< Operator

4.42.4.3 istream & operator>> (istream & *is*, CNetzplanKante & *Zielkante*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CNetzplanKante.h**
- **CNetzplanKante.cpp**

4.43 CNetzplanKnoten Klassenreferenz

`#include <CNetzplanKnoten.h>`

Öffentliche Datenelemente

- **CNetzplanKnoten** ()
- **CNetzplanKnoten** (int eineId, int einePrioritaet)
- **CNetzplanKnoten** (const CNetzplanKnoten&)
- CNetzplanKnoten& **operator=** (const CNetzplanKnoten&)
- void **setzenID** (int eineID)
- void **setzenPrioritaet** (int einePrioritaet)
- int **holenID** ()
- int **holenPrioritaet** ()
- edge_map<int>* **holenKantenListe** ()
- **LEDA_MEMORY** (CNetzplanKnoten)

Freundbeziehungen

- istream& **operator>>** (istream&, CNetzplanKnoten&)
- ostream& **operator<<** (ostream&, const CNetzplanKnoten&)
- int **compare** (const CNetzplanKnoten&, const CNetzplanKnoten&)

4.43.1 Ausführliche Beschreibung

Knotenojekt des CNetzplans Enthaelt die Attribute ID und Prioritaet und jeweilige setzen- und geben-Operationen

4.43.2 Beschreibung der Konstruktoren und Destruktoren

4.43.2.1 CNetzplanKnoten::CNetzplanKnoten ()

Default Konstruktor

4.43.2.2 CNetzplanKnoten::CNetzplanKnoten (int *eineId*, int *eine-Prioritaet*)

Konstruktor mit initialisierung

4.43.2.3 CNetzplanKnoten::CNetzplanKnoten (const CNetzplan-Knoten & *originalKnoten*)

Kopier Konstruktor

4.43.3 Dokumentation der Elementfunktionen

4.43.3.1 CNetzplanKnoten::LEDA_MEMORY (CNetzplanKnoten)

4.43.3.2 int CNetzplanKnoten::holenID ()

4.43.3.3 edge_map< int >* CNetzplanKnoten::holenKantenListe ()

4.43.3.4 int CNetzplanKnoten::holenPrioritaet ()

4.43.3.5 CNetzplanKnoten & CNetzplanKnoten::operator= (const CNetzplanKnoten & *originalKnoten*)

= Operator

4.43.3.6 void CNetzplanKnoten::setzenID (int *eineID*)

4.43.3.7 void CNetzplanKnoten::setzenPrioritaet (int *eine-Prioritaet*)

4.43.4 Freundbeziehungen und Funktionsdokumentation

4.43.4.1 int compare (const CNetzplanKnoten & *knoten1*, const CNetzplanKnoten & *knoten2*) [friend]

compare Operator

4.43.4.2 ostream & operator<< (ostream & *os*, const CNetzplanKnoten & *ausgabeKante*) [friend]

<< Operator

4.43.4.3 istream & operator>> (istream & *is*, CNetzplanKnoten & *zielKante*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CNetzplanKnoten.h**
- **CNetzplanKnoten.cpp**

4.44 CParameterHaltestelle Klassenreferenz

`#include <CParameterHaltestelle.h>`

Öffentliche Datenelemente

- **CParameterHaltestelle** (list<list<**CPunkt***>*> * Strassennetzliste, list<**CPunkt***> * Pflichthaltestellen, list<list<**CPunkt***>*> * Wege, list<**CBevoelkerungsdichteElement***> * Bevoelkerungsdichtenliste,matrix * QuelleZielMatrix,list<**CQuelleZielListeElement***> * QuelleZielListe,matrix * Pendlermatrix,list<**CPendlerListeElement***> * Pendlerliste, list<list<**CPunkt** *> *> * Gebaeudeliste, **TAlgorithmus** Algorithmustyp=stochastisch,int Zufallszahlengeneratorinitialisierung=-1,double minimalesEinzugsgebiet=150,double maximalesEinzugsgebiet=450,double verbindungsradius=1200,**TRadius** Radiustyp=luftlinie,double Haltestellenlage=0,double toleranzgrenze=0.5,unsigned int BevoelkerungsrasterEinheiten=10,double koeffizientBevoelkerungsueberversorgung=0.5,double koeffizientBevoelkerungsunterversorgung=0.5,double koeffizientFlaechenueberversorgung=0.5,double koeffizientFlaechenunterversorgung=0.5,double koeffizientFahrtwunscheinbeziehung=0.5,double koeffizientStrassenanpassung=0.5, bool AnzeigenLegende = true, bool AnzeigenStrassennetz = true, bool AnzeigenBevoelkerung = true, bool AnzeigenFlaeche = true)
- **~CParameterHaltestelle** ()
- void **laden** (const char * Dateiname)

- void **speichern** (const char * Dateiname)
- **TAlgorithmus holenAlgorithmustyp** (void)
- void **setzenAlgorithmustyp** (**TAlgorithmus** Algorithmustyp)
- int **holenZufallszahlengeneratorinitialisierung** (void)
- void **setzenZufallszahlengeneratorinitialisierung** (int Zufallszahlengeneratorinitialisierung)
- double **holenMinimalesEinzugsgebiet** (void)
- void **setzenMinimalesEinzugsgebiet** (double minimalesEinzugsgebiet)
- double **holenMaximalesEinzugsgebiet** (void)
- void **setzenMaximalesEinzugsgebiet** (double maximales-Einzugsgebiet)
- double **holenVerbindungsradius** (void)
- void **setzenVerbindungsradius** (double verbindungsradius)
- **TRadius holenRadiustyp** (void)
- void **setzenRadiustyp** (**TRadius** Radiustyp)
- double **holenHaltestellenlage** (void)
- void **setzenHaltestellenlage** (double Haltestellenlage)
- double **holenToleranzgrenze** (void)
- void **setzenToleranzgrenze** (double toleranzgrenze)
- list<**CPunkt***>* **holenPflichthaltestellen** (void)
- void **setzenPflichthaltestellen** (list<**CPunkt***> * Pflichthaltestellen)
- list<list<**CPunkt***>*>* **holenStrassennetzliste** (void)
- void **setzenStrassennetzliste** (list<list<**CPunkt***>*> * Strassennetzliste)
- list<list<**CPunkt***>*>* **holenWegeliste** (void)
- void **setzenWegeliste** (list<list<**CPunkt***>*> * wege)
- unsigned int **holenBevoelkerungsrasterEinheiten** (void)
- void **setzenBevoelkerungsrasterEinheiten** (unsigned int BevoelkerungsrasterEinheiten)
- double **holenKoeffizientBevoelkerungsueberversorgung** (void)
- void **setzenKoeffizientBevoelkerungsueberversorgung** (double koeffizientBevoelkerungsueberversorgung)
- double **holenKoeffizientBevoelkerungsunterversorgung** (void)
- void **setzenKoeffizientBevoelkerungsunterversorgung** (double koeffizientBevoelkerungsunterversorgung)
- double **holenKoeffizientFlaechenueberversorgung** (void)
- void **setzenKoeffizientFlaechenueberversorgung** (double koeffizient-Flaechenueberversorgung)
- double **holenKoeffizientFlaechenunterversorgung** (void)
- void **setzenKoeffizientFlaechenunterversorgung** (double koeffizient-Flaechenunterversorgung)
- double **holenKoeffizientFahrtwunscheinbeziehung** (void)
- void **setzenKoeffizientFahrtwunscheinbeziehung** (double koeffizient-Fahrtwunscheinbeziehung)
- double **holenKoeffizientStrassenanpassung** (void)
- void **setzenKoeffizientStrassenanpassung** (double koeffizient-Strassenanpassung)

- bool **pruefenAnzeigenLegende** (void)
- void **setzenAnzeigenLegende** (bool anzeigen)
- bool **pruefenAnzeigenStrassennetz** (void)
- void **setzenAnzeigenStrassennetz** (bool anzeigen)
- bool **pruefenAnzeigenBevoelkerung** (void)
- void **setzenAnzeigenBevoelkerung** (bool anzeigen)
- bool **pruefenAnzeigenFlaeche** (void)
- void **setzenAnzeigenFlaeche** (bool anzeigen)
- bool **pruefenBatchModus** (void)
- void **setzenBatchModus** (bool istbatchmodus)
- **TStrategie pruefenStrategie** (void)
- void **setzenStrategie** (**TStrategie** strategie)
- unsigned int **pruefenAnzahlEltern** (void)
- void **setzenAnzahlEltern** (unsigned int anzahl)
- unsigned int **pruefenAnzahlKinder** (void)
- void **setzenAnzahlKinder** (unsigned int anzahl)
- unsigned int **pruefenAnzahlGenerationen** (void)
- void **setzenAnzahlGenerationen** (unsigned int anzahl)
- bool **pruefenWerteunterschreitung** (void)
- void **setzenWerteunterschreitung** (bool aktiv)
- double **pruefenGrenzwert** (void)
- void **setzenGrenzwert** (double wert)
- unsigned int **holenAnzahlHaltestellen** (void)
- void **setzenAnzahlHaltestellen** (unsigned int anzahl)
- list<**CBevoelkerungsdichteElement**∗>∗ **holenBevoelkerungsdichtenliste** (void)
- void **setzenBevoelkerungsdichtenliste** (list<**CBevoelkerungsdichteElement**∗> ∗ Bevoelkerungsdichtenliste)
- matrix∗ **holenQuelleZielMatrix** (void)
- void **setzenQuelleZielMatrix** (matrix ∗ QuelleZielMatrix)
- list<**CQuelleZielListeElement**∗>∗ **holenQuelleZielListe** (void)
- void **setzenQuelleZielListe** (list<**CQuelleZielListeElement**∗> ∗ QuelleZielListe)
- matrix∗ **holenPendlermatrix** (void)
- void **setzenPendlermatrix** (matrix ∗ Pendlermatrix)
- list<**CPendlerListeElement**∗ >∗ **holenPendlerliste** (void)
- void **setzenPendlerliste** (list<**CPendlerListeElement**∗ > ∗ Pendlerliste)
- list<list<**CPunkt**∗ > ∗ >∗ **holenGebaeudeliste** (void)
- void **setzenGebaeudeliste** (list<list<**CPunkt**∗ > ∗ > ∗ Gebaeudeliste)

4.44.1 Ausführliche Beschreibung

Dient der Aufnahme aller Parameter, welche fuer das Modul Haltestelle bestimmt sind. Dient der einfacheren Uebergabe von Werten.

4.44.2 Beschreibung der Konstruktoren und Destruktoren

4.44.2.1 CParameterHaltestelle::CParameterHaltestelle (list< list< CPunkt *>*>* *Strassennetzliste*, list< CPunkt *>* *Pflichthaltestellen*, list< list< CPunkt *>*>* *Wege*, list< CBevoelkerungsdichteElement *>* *Bevoelkerungsdichtenliste*, matrix * *QuelleZielMatrix*, list< CQuelleZielListeElement *>* *QuelleZielListe*, matrix * *Pendlermatrix*, list< CPendlerListeElement * >* *Pendlerliste*, list< list< CPunkt *>*>* *Gebaeudeliste*, TAlgorithmus *Algorithmustyp* = stochastisch, int *Zufallszahlengeneratorinitialisierung* = -1, double *minimalesEinzugsgebiet* = 150, double *maximalesEinzugsgebiet* = 450, double *verbindungsradius* = 1200, TRadius *Radiustyp* = luftlinie, double *Haltestellenlage* = 0, double *toleranzgrenze* = 0.5, unsigned int *BevoelkerungsrasterEinheiten* = 10, double *koeffizientBevoelkerungsueberversorgung* = 0.5, double *koeffizientBevoelkerungsunterversorgung* = 0.5, double *koeffizientFlaechenueberversorgung* = 0.5, double *koeffizientFlaechenunterversorgung* = 0.5, double *koeffizientFahrtwunscheinbeziehung* = 0.5, double *koeffizientStrassenanpassung* = 0.5, bool *AnzeigenLegende* = true, bool *AnzeigenStrassennetz* = true, bool *AnzeigenBevoelkerung* = true, bool *AnzeigenFlaeche* = true)

Standard-Konstruktor. Belegt alle Variablen mit definierten Vorgabewerten.

4.44.2.2 CParameterHaltestelle::~CParameterHaltestelle ()

4.44.3 Dokumentation der Elementfunktionen

4.44.3.1 TAlgorithmus CParameterHaltestelle::holenAlgorithmustyp (void)

4.44.3.2 unsigned int CParameterHaltestelle::holenAnzahlHaltestellen (void)

4.44.3.3 list< CBevoelkerungsdichteElement *>* CParameterHaltestelle::holenBevoelkerungsdichtenliste (void)

4.44.3.4 unsigned int CParameterHaltestelle::holenBevoelkerungsrasterEinheiten (void)

4.44.3.5 list< list< CPunkt * >* >* CParameterHaltestelle::holenGebaeudeliste (void)

4.44.3.6 double CParameterHaltestelle::holenHaltestellenlage (void)

4.44.3.7 double CParameterHaltestelle::holenKoeffizient-Bevoelkerungsueberversorgung (void)

4.44.3.8 double CParameterHaltestelle::holenKoeffizient-Bevoelkerungsunterversorgung (void)

4.44.3.9 double CParameterHaltestelle::holenKoeffizient-Fahrtwunscheinbeziehung (void)

4.44.3.10 double CParameterHaltestelle::holenKoeffizient-Flaechenueberversorgung (void)

4.44.3.11 double CParameterHaltestelle::holenKoeffizient-Flaechenunterversorgung (void)

4.44.3.12 double CParameterHaltestelle::holenKoeffizient-Strassenanpassung (void)

4.44.3.13 double CParameterHaltestelle::holenMaximales-Einzugsgebiet (void)

4.44.3.14 double CParameterHaltestelle::holenMinimales-Einzugsgebiet (void)

4.44.3.15 list< CPendlerListeElement * >* CParameter-Haltestelle::holenPendlerliste (void)

4.44.3.16 matrix * CParameterHaltestelle::holenPendlermatrix (void)

4.44.3.17 list< CPunkt *>* CParameterHaltestelle::holen-Pflichthaltestellen (void)

4.44.3.18 list< CQuelleZielListeElement *>* CParameter-Haltestelle::holenQuelleZielListe (void)

4.44.3.19 matrix * CParameterHaltestelle::holenQuelleZielMatrix (void)

4.44.3.20 TRadius CParameterHaltestelle::holenRadiustyp (void)

Ermittelt den Modus fuer die Ermittlung der von Entfernungen zugrundeliegenden Berechnungsart.

4.44.3.21 list< list< CPunkt *>*>* CParameterHaltestelle::holenStrassennetzliste (void)

4.44.3.22 double CParameterHaltestelle::holenToleranzgrenze (void)

4.44.3.23 double CParameterHaltestelle::holenVerbindungsradius (void)

4.44.3.24 list< list< CPunkt *>*>* CParameterHaltestelle::holenWegeliste (void)

4.44.3.25 int CParameterHaltestelle::holenZufallszahlengeneratorinitialisierung (void)

4.44.3.26 void CParameterHaltestelle::laden (const char * *Dateiname*)

Laedt eine zuvor gespeicherte Konfiguration mit dem angegebenen Dateinamen.

4.44.3.27 unsigned int CParameterHaltestelle::pruefenAnzahlEltern (void)

4.44.3.28 unsigned int CParameterHaltestelle::pruefenAnzahlGenerationen (void)

4.44.3.29 unsigned int CParameterHaltestelle::pruefenAnzahlKinder (void)

4.44.3.30 bool CParameterHaltestelle::pruefenAnzeigenBevoelkerung (void)

4.44.3.31 bool CParameterHaltestelle::pruefenAnzeigenFlaeche (void)

4.44.3.32 bool CParameterHaltestelle::pruefenAnzeigenLegende (void)

4.44.3.33 bool CParameterHaltestelle::pruefenAnzeigenStrassennetz (void)

4.44.3.34 bool CParameterHaltestelle::pruefenBatchModus (void)

4.44.3.35 double CParameterHaltestelle::pruefenGrenzwert (void)

4.44.3.36 TStrategie CParameterHaltestelle::pruefenStrategie (void)

4.44.3.37 bool CParameterHaltestelle::pruefenWerteunterschreitung (void)

4.44.3.38 void CParameterHaltestelle::setzenAlgorithmustyp (TAlgorithmus *Algorithmustyp*)

4.44.3.39 void CParameterHaltestelle::setzenAnzahlEltern (unsigned int *anzahl*)

4.44.3.40 void CParameterHaltestelle::setzenAnzahlGenerationen (unsigned int *anzahl*)

4.44.3.41 void CParameterHaltestelle::setzenAnzahlHaltestellen (unsigned int *anzahl*)

4.44.3.42 void CParameterHaltestelle::setzenAnzahlKinder (unsigned int *anzahl*)

4.44.3.43 void CParameterHaltestelle::setzenAnzeigenBevoelkerung (bool *anzeigen*)

4.44.3.44 void CParameterHaltestelle::setzenAnzeigenFlaeche (bool *anzeigen*)

4.44.3.45 void CParameterHaltestelle::setzenAnzeigenLegende (bool *anzeigen*)

4.44.3.46 void CParameterHaltestelle::setzenAnzeigenStrassennetz (bool *anzeigen*)

4.44.3.47 void CParameterHaltestelle::setzenBatchModus (bool *istbatchmodus* = false)

4.44.3.48 void CParameterHaltestelle::setzenBevoelkerungsdichtenliste (list< CBevoelkerungsdichteElement *>* *Bevoelkerungsdichtenliste*)

4.44.3.49 void CParameterHaltestelle::setzenBevoelkerungsrasterEinheiten (unsigned int *BevoelkerungsrasterEinheiten*)

4.44.3.50 void CParameterHaltestelle::setzenGebaeudeliste (list< list< CPunkt * >* >* *Gebaeudeliste*)

4.44.3.51 void CParameterHaltestelle::setzenGrenzwert (double *wert*)

4.44.3.52 void CParameterHaltestelle::setzenHaltestellenlage (double *Haltestellenlage*)

4.44.3.53 void CParameterHaltestelle::setzenKoeffizientBevoelkerungsueberversorgung (double *koeffizientBevoelkerungsueberversorgung*)

4.44.3.54 void CParameterHaltestelle::setzenKoeffizientBevoelkerungsunterversorgung (double *koeffizientBevoelkerungsunterversorgung*)

4.44.3.55 void CParameterHaltestelle::setzenKoeffizientFahrtwunscheinbeziehung (double *koeffizientFahrtwunscheinbeziehung*)

4.44.3.56 void CParameterHaltestelle::setzenKoeffizientFlaechenueberversorgung (double *koeffizientFlaechenueberversorgung*)

4.44.3.57 void CParameterHaltestelle::setzenKoeffizientFlaechenunterversorgung (double *koeffizientFlaechenunterversorgung*)

4.44.3.58 void CParameterHaltestelle::setzenKoeffizientStrassenanpassung (double *koeffizientStrassenanpassung*)

4.44.3.59 void CParameterHaltestelle::setzenMaximalesEinzugsgebiet (double *maximalesEinzugsgebiet*)

4.44.3.60 void CParameterHaltestelle::setzenMinimales-Einzugsgebiet (double *minimalesEinzugsgebiet*)

4.44.3.61 void CParameterHaltestelle::setzenPendlerliste (list< CPendlerListeElement * >* *Pendlerliste*)

4.44.3.62 void CParameterHaltestelle::setzenPendlermatrix (matrix * *Pendlermatrix*)

4.44.3.63 void CParameterHaltestelle::setzenPflichthaltestellen (list< CPunkt *>* *Pflichthaltestellen*)

4.44.3.64 void CParameterHaltestelle::setzenQuelleZielListe (list< CQuelleZielListeElement *>* *QuelleZielListe*)

4.44.3.65 void CParameterHaltestelle::setzenQuelleZielMatrix (matrix * *QuelleZielMatrix*)

4.44.3.66 void CParameterHaltestelle::setzenRadiustyp (TRadius *Radiustyp*)

4.44.3.67 void CParameterHaltestelle::setzenStrassennetzliste (list< list< CPunkt *>*>* *Strassennetzliste*)

4.44.3.68 void CParameterHaltestelle::setzenStrategie (TStrategie *strategie*)

4.44.3.69 void CParameterHaltestelle::setzenToleranzgrenze (double *toleranzgrenze*)

4.44.3.70 void CParameterHaltestelle::setzenVerbindungsradius (double *verbindungsradius*)

4.44.3.71 void CParameterHaltestelle::setzenWegeliste (list< list< CPunkt *>*>* *wege*)

4.44.3.72 void CParameterHaltestelle::setzenWerteunterschreitung (bool *aktiv*)

4.44.3.73 void CParameterHaltestelle::setzen-Zufallszahlengeneratorinitialisierung (int *Zufallszahlengeneratorinitialisierung*)

4.44.3.74 void CParameterHaltestelle::speichern (const char * *Dateiname*)

Speichert eine Konfiguration unter dem angegebenen Dateinamen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CParameterHaltestelle.h**
- **CParameterHaltestelle.cpp**

4.45 CPendlerListeElement Klassenreferenz

`#include <CPendlerListeElement.h>`

Öffentliche Datenelemente

- double **holenRadius** ()
- **CPunkt*** **holenPunkt** ()
- int **holenID** ()
- void **setzenRadius** (double r)
- void **setzenPunkt** (**CPunkt** *EinPunkt)
- void **setzenID** (int eineID)
- **CPendlerListeElement** (**CPunkt** * einPunkt, int initid, double initradius)
- **~CPendlerListeElement** ()

4.45.1 Ausführliche Beschreibung

Die PendlerListe dient dazu die Indices aus der Pendlermatrix auf Koordinaten umzulegen. In einem Element befindet sich die id der einzelen Matrixspalten, der Punkt, auf den sich der Index bezieht, und ein Radius, der die Ausdehung und somit den Gültigkeitsbereich des Wertes angibt.

4.45.2 Beschreibung der Konstruktoren und Destruktoren

4.45.2.1 CPendlerListeElement::CPendlerListeElement (CPunkt * *einPunkt*, int *initid*, double *initradius*)

Konstruktor

4.45.2.2 CPendlerListeElement::~CPendlerListeElement ()

Destruktor

4.45.3 Dokumentation der Elementfunktionen

4.45.3.1 int CPendlerListeElement::holenID ()

Methode um die ID zu holen

4.45.3.2 CPunkt * CPendlerListeElement::holenPunkt ()

Methode um den Punkt zu holen

4.45.3.3 double CPendlerListeElement::holenRadius ()

Methode um den radius zu holen

4.45.3.4 void CPendlerListeElement::setzenID (int *eineID*)

Methode um die ID eines Elementes zu setzen

4.45.3.5 void CPendlerListeElement::setzenPunkt (CPunkt * *EinPunkt*)

Methode um den Punkt zu setzen

4.45.3.6 void CPendlerListeElement::setzenRadius (double *r*)

Methode um den radius zu setzen

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CPendlerListeElement.h**
- **CPendlerListeElement.cpp**

4.46 CPlanquadrat Klassenreferenz

`#include <CPlanquadrat.h>`

Öffentliche Datenelemente

- void **setzen** (unsigned int X, unsigned int Y)
- unsigned int **holenX** (void)
- unsigned int **holenY** (void)
- list<CStrasse*> **holenStrassen** (void)
- void **hinzufuegenStrasse** (CStrasse * Strasse)
- void **loeschenStrassen** (void)
- void **loeschenStrasse** (CStrasse * Strasse)

4.46.1 Dokumentation der Elementfunktionen

4.46.1.1 void CPlanquadrat::hinzufuegenStrasse (CStrasse * *Strasse*)

Fuegt eine Strasse diesem Planquadrat hinzu.

4.46.1.2 list< CStrasse *> CPlanquadrat::holenStrassen (void)

Liefert alle Strassen zurueck, welche das Planquadrat beruehren.

4.46.1.3 unsigned int CPlanquadrat::holenX (void)

Holt die X-ID eines Planquadrates.

4.46.1.4 unsigned int CPlanquadrat::holenY (void)

Holt die Y-ID eines Planquadrates.

4.46.1.5 void CPlanquadrat::loeschenStrasse (CStrasse * *Strasse*)

Entfernt die uebergebene Strasse aus dem Planquadrat.

4.46.1.6 void CPlanquadrat::loeschenStrassen (void)

Loescht alle Strassen aus dem Planquadrat.

4.46.1.7 void CPlanquadrat::setzen (unsigned int *X*, unsigned int *Y*)

Bestimmt die ID eines Planquadrates.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CPlanquadrat.h**
- **CPlanquadrat.cpp**

4.47 CPunkt Klassenreferenz

`#include <CPunkt.h>`

Öffentliche Datenelemente

- **CPunkt** (double xKoordinate, double yKoordinate)
- void **setzen** (double xKoordinate, double yKoordinate)
- void **holen** (double * xKoordinate, double * yKoordinate)
- void **holen** (double& x, double& y)
- double **holenX** ()
- double **holenY** ()
- **CPunkt** (void)
- **CPunkt** (const CPunkt& originalBeschriftung)
- **~CPunkt** ()

Freundbeziehungen

- ostream& **operator<<** (ostream&, const CPunkt&)

4.47.1 Ausführliche Beschreibung

Start- und Zielpunkte werden hierüber definiert.

4.47.2 Beschreibung der Konstruktoren und Destruktoren

4.47.2.1 CPunkt::CPunkt (double *xKoordinate*, double *yKoordinate*)

Konstruktor, welcher das Setzen der Koordinaten uebernimmt.

Parameter:

xKoordinate X-Koordinate des zu erzeugenden Punktes.

yKoordinate Y-Koordinate des zu erzeugenden Punktes.

4.47.2.2 CPunkt::CPunkt (void)

Konstruktor. Setzt die Position auf (0/0).

4.47.2.3 CPunkt::CPunkt (const CPunkt & *originalBeschriftung*)

Kopierkonstruktor

4.47.2.4 CPunkt::~CPunkt ()

Destruktor

4.47.3 Dokumentation der Elementfunktionen

4.47.3.1 void CPunkt::holen (double & *x*, double & *y*)

Gibt das Koordinatenpaar des Punktes zurueck

Parameter:

xKoordinate X-Koordinate des zu ermittelnden Punktes.

yKoordinate Y-Koordinate des zu ermittelnden Punktes.

4.47.3.2 void CPunkt::holen (double * *xKoordinate*, double * *yKoordinate*)

Gibt das Koordinatenpaar des Punktes als Zeiger zurueck

Parameter:

xKoordinate X-Koordinate des zu ermittelnden Punktes.

yKoordinate Y-Koordinate des zu ermittelnden Punktes.

4.47.3.3 double CPunkt::holenX ()

Gibt den X-Wert des Punktes zurueck

4.47.3.4 double CPunkt::holenY ()

Gibt den Y-Wert des Punktes zurueck

4.47.3.5 void CPunkt::setzen (double *xKoordinate*, double *yKoordinate*)

Setze das Koordinatenpaar des Punktes

Parameter:

xKoordinate X-Koordinate des zu setzenden Punktes.

yKoordinate Y-Koordinate des zu setzenden Punktes.

4.47.4 Freundbeziehungen und Funktionsdokumentation

4.47.4.1 ostream & operator<< (ostream & *os*, const CPunkt & *beschriftung*) [friend]

<< Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CPunkt.h**
- **CPunkt.cpp**

4.48 CQuelleZielListeElement Klassenreferenz

`#include <CQuelleZielListeElement.h>`

Öffentliche Datenelemente

- **CPunkt* holenPunkt** ()
- int **holenID** ()
- void **setzenPunkt** (**CPunkt*** EinPunkt)
- void **seztenID** (int i)
- **CQuelleZielListeElement** (**CPunkt** *EinPunkt, int i)
- **~CQuelleZielListeElement** ()

4.48.1 Ausführliche Beschreibung

Diese Klasse dient dazu, die Indizes der Spalten der QuelleZielMatrix auf die Koordinaten abzubilden

4.48.2 Beschreibung der Konstruktoren und Destruktoren

4.48.2.1 CQuelleZielListeElement::CQuelleZielListeElement (CPunkt * *EinPunkt*, int *i*)

Konstruktor

4.48.2.2 CQuelleZielListeElement::~CQuelleZielListeElement ()

Destruktor

4.48.3 Dokumentation der Elementfunktionen

4.48.3.1 int CQuelleZielListeElement::holenID ()

holen Methode für die ID

4.48.3.2 CPunkt * CQuelleZielListeElement::holenPunkt ()

holen Methode für den Punkt

4.48.3.3 void CQuelleZielListeElement::setzenPunkt (CPunkt * *EinPunkt*)

Methode um den Punkt zu setzen

4.48.3.4 void CQuelleZielListeElement::seztenID (int *i*)

Methode um die ID zu setzen

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CQuelleZielListeElement.h**
- **CQuelleZielListeElement.cpp**

4.49 CReisebewertung Klassenreferenz

`#include <CReisebewertung.h>`

Öffentliche Datenelemente

- **CReisebewertung** ()
- **~CReisebewertung** ()
- **CReisebewertung** (const CReisebewertung&)
- CReisebewertung& **operator=** (const CReisebewertung&)
- void **setzenFahrzeit** (int)
- int **gebenFahrzeit** ()
- void **setzenWartezeit** (int)

- int **gebenWartezeit** ()
- void **setzenAnzahlDerUmstiege** (int)
- int **gebenAnzahlDerUmstiege** ()
- void **setzenIdealeFahrzeit** (double)
- double **gebenIdealeFahrzeit** ()
- void **setzenReiseweg** (list<three_tuple<int,int,int> >)
- list<three_tuple<int,int,int> > **gebenReiseweg** ()

Freundbeziehungen

- istream& **operator>>** (istream&, CReisebewertung&)
- ostream& **operator<<** (ostream&, const CReisebewertung&)

4.49.1 Ausführliche Beschreibung

Bei dieser Klasse handelt es sich um eine Sammelstelle fuer Fahrplan Guetemasse

4.49.2 Beschreibung der Konstruktoren und Destruktoren

4.49.2.1 CReisebewertung::CReisebewertung ()

Default Konstruktor

4.49.2.2 CReisebewertung::~CReisebewertung ()

Default Destruktor

4.49.2.3 CReisebewertung::CReisebewertung (const CReisebewertung & *reisebewertungOriginal*)

Kopier-Konstruktor

4.49.3 Dokumentation der Elementfunktionen

4.49.3.1 int CReisebewertung::gebenAnzahlDerUmstiege ()

4.49.3.2 int CReisebewertung::gebenFahrzeit ()

4.49.3.3 double CReisebewertung::gebenIdealeFahrzeit ()

4.49.3.4 list< three_tuple< int,int,int > > CReisebewertung::gebenReiseweg ()

4.49.3.5 int CReisebewertung::gebenWartezeit ()

4.49.3.6 CReisebewertung & CReisebewertung::operator= (const CReisebewertung & *reisebewertungOriginal*)

= Operator

4.49.3.7 void CReisebewertung::setzenAnzahlDerUmstiege (int *umstiege*)

4.49.3.8 void CReisebewertung::setzenFahrzeit (int *fahrzeit*)

4.49.3.9 void CReisebewertung::setzenIdealeFahrzeit (double *idealeFahrzeit*)

4.49.3.10 void CReisebewertung::setzenReiseweg (list< three_tuple< int,int,int > > *reiseweg*)

4.49.3.11 void CReisebewertung::setzenWartezeit (int *wartezeit*)

4.49.4 Freundbeziehungen und Funktionsdokumentation

4.49.4.1 ostream & operator<< (ostream & *outstream*, const CReisebewertung & *reisebewertung*) [friend]

<< Operator

4.49.4.2 istream & operator>> (istream & *instream*, CReisebewertung & *reisebewertung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CReisebewertung.h**
- **CReisebewertung.cpp**

4.50 CSchnittstelle Klassenreferenz

`#include <CSchnittstelle.h>`

Klassendiagramm für CSchnittstelle:

Öffentliche Datenelemente

- **CSchnittstelle** (**CDatenbank**∗ datenbank)

Geschützte Attribute

- **CKontrolle**∗ **lnkCKontrolle**
- **CDatenbank**∗ **lnkCDatenbank**

4.50.1 Ausführliche Beschreibung

Dies ist die abstrakte Superklasse fuer alle Schnittstellenklassen.

4.50.2 Beschreibung der Konstruktoren und Destruktoren

4.50.2.1 CSchnittstelle::CSchnittstelle (CDatenbank ∗ *datenbank*) [inline]

4.50.3 Dokumentation der Datenelemente

4.50.3.1 CDatenbank ∗ CSchnittstelle::lnkCDatenbank [protected]

@supplierCardinality 1

4.50.3.2 CKontrolle ∗ CSchnittstelle::lnkCKontrolle [protected]

@supplierCardinality 1

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CSchnittstelle.h**

4.51 CSchnittstelleMitGUI Klassenreferenz

`#include <CSchnittstelleMitGUI.h>`

Klassendiagramm für CSchnittstelleMitGUI:

Öffentliche Datenelemente

- **CSchnittstelleMitGUI** (**CDatenbank**∗, **CGUI**∗)
- void **meldenLaufzeitAusgabe** (char∗)
- void **anzeigenMeldung** (char∗)

Geschützte Attribute

- **CGUI**∗ **lnkCGUI**

4.51.1 Ausführliche Beschreibung

Stellt die Kommunikation zwischen der GUI, der Datenbank und den anderen modulspezifischen Klassen her. Von dieser abstrakten Superklasse werden die Schnittstellen der einzelnen Module abgeleitet.

4.51.2 Beschreibung der Konstruktoren und Destruktoren

4.51.2.1 CSchnittstelleMitGUI::CSchnittstelleMitGUI (CDatenbank ∗ *datenbank*, CGUI ∗ *gui*)

4.51.3 Dokumentation der Elementfunktionen

4.51.3.1 void CSchnittstelleMitGUI::anzeigenMeldung (char ∗)

veranlasst das Oeffnen eines Fensters mit einer Meldung

4.51.3.2 void CSchnittstelleMitGUI::meldenLaufzeitAusgabe (char ∗ *text*)

meldet Laufzeit-Ausgaben an die Gui

4.51.4 Dokumentation der Datenelemente

4.51.4.1 CGUI ∗ CSchnittstelleMitGUI::lnkCGUI `[protected]`

@supplierCardinality 1

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSchnittstelleMitGUI.h**
- **CSchnittstelleMitGUI.cpp**

4.52 CSchnittstellenTest Klassenreferenz

`#include <CSchnittstellenTest.h>`

Öffentliche Datenelemente

- **~CSchnittstellenTest** ()
- void **fahrplanerstellungWegenetzplan** ()
- int **fahrplanerstellungMaximaleFahrzeuganzahl** ()
- int **fahrplanerstellungAllgemeinerTakt** ()
- void **fahrplanerstellungErgebnisrueckgabe** (list<**CFahrplanElement**∗>∗)
- void **fahrplanerstellungWartezeiten** (double, double, double, double)
- void **fahrplanerstellungAnzahlBenoetigterFahrzeuge** (int)
- void **fahrplanerstellungHeuristischeLoesung** (int)
- void **fahrplanerstellungZusammenhangskomponenten** (array<list<three_tuple<int,int,int>∗ >∗ > ∗)
- array<three_tuple<int,int,int>∗ >∗ **fahrplanerstellungAbfahrtszeiten** ()
- void **fahrplanerstellungMenueaktivierung** ()
- h_array<int,int>∗ **fahrplanerstellungMindesthaltezeit** ()
- h_array<int,list<three_tuple<int,int,int>∗ >∗>∗ **fahrplanerstellungMindestumsteigezeit** ()
- void **fahrplanerstellungAusgabeFahrplan** ()
- void **fahrplanerstellungBerechnenFahrzeugAnzahl** ()
- void **fahrplanerstellungBerechnenWartezeiten** ()
- four_tuple<int,int,int,int>∗ **fahrplanerstellungFahrplanIntervall** ()
- list<four_tuple<int, int, int, int>∗ >∗ **fahrplanerstellungFesteVerbindungen** ()
- void **fahrplanerstellungEntscheidungsbaumverfahren** (int,int)
- void **fahrplanerstellungSukzessiveLoesungsverbesserung** (int)
- void **speichernGuetemasse** (**CGuetemasse**∗)
- **CGuetemasse**∗ **holenGuetemasse** ()

4.52.1 Ausführliche Beschreibung

Diese Klasse dient als Schnittstellentest

4.52.2 Beschreibung der Konstruktoren und Destruktoren

4.52.2.1 CSchnittstellenTest::~CSchnittstellenTest ()

4.52.3 Dokumentation der Elementfunktionen

4.52.3.1 array< three_tuple< int,int,int >* >* CSchnittstellenTest::fahrplanerstellungAbfahrtszeiten ()

Die Methode liefert ein Array von four_tuple zurueck, welche folgende Werte haben:

- HaltestellenID
- Liniennr
- minute (Reicht aus, da in dem Intervall ein fester Takt gilt);

Das Array muss dabei die gleiche Reihenfolge haben, wie das Array der Zusammenhangskomponenten

4.52.3.2 int CSchnittstellenTest::fahrplanerstellungAllgemeinerTakt ()

Bei dieser Methode wird der Allgemeine Takt (Minuten - Wert) uebergeben.

4.52.3.3 void CSchnittstellenTest::fahrplanerstellungAnzahlBenoetigterFahrzeuge (int *fahrzeugAnzahl*)

Uebergabe der benoetigten Fahrzeug-Anzahl

4.52.3.4 void CSchnittstellenTest::fahrplanerstellungAusgabeFahrplan ()

Diese Methode ruft im Fahrplan die Methode ausgebenFahrplan auf. Nach Beendigung der Methode wurde ein Fahrplan uebergeben

4.52.3.5 void CSchnittstellenTest::fahrplanerstellungBerechnenFahrzeugAnzahl ()

4.52.3.6 void CSchnittstellenTest::fahrplanerstellungBerechnenWartezeiten ()

4.52.3.7 void CSchnittstellenTest::fahrplanerstellungEntscheidungsbaumverfahren (int *zusammenhangskomponentenID*, int *baumtiefe*)

Starten der Berechnung des Entscheidungsbaumverfahrens der entsprechenden Zusammenhangskomponete gekennzeichnet durch die ID

4.52.3.8 void CSchnittstellenTest::fahrplanerstellungErgebnisrueckgabe (list< CFahrplanElement *>* *fahrplan*)

Diese Methode erhaelt den erstellten Fahrplan zur Speicherung in die Datenbank.

4.52.3.9 four_tuple< int,int,int,int >* CSchnittstellenTest::fahrplanerstellungFahrplanIntervall ()

Bei dieser Methode wird das Intervall zurueckgegeben, fuer das der Fahrplan gueltig sein soll Dabei gilt folgende Reihenfolge bei den int - Werten : Stunde Von, Minute Von, Stunde Bis, Minute Bis

4.52.3.10 list< four_tuple< int,int,int,int >* >* CSchnittstellenTest::fahrplanerstellungFesteVerbindungen ()

Setzen der Liste von festen Verbindungen Diese Methode liefert in einer Liste alle tupel zurueck die eine feste Bindung haben sollen. Dabei wird die ZusammenhangskomponentenID, die ID der Linie 1, die ID der 2. Linie und die Umsteigehaltestellen ID uebergeben.

4.52.3.11 void CSchnittstellenTest::fahrplanerstellungHeuristischeLoesung (int *zusammenhangskomponentenID*)

Starten der Berechnung einer Ausgangsloesung fuer die entsprechende Zusammenhangskomponete gekennzeichnet durch die ID

4.52.3.12 int CSchnittstellenTest::fahrplanerstellungMaximaleFahrzeuganzahl ()

Hier soll die Maximale Fahrzeuganzahl uebergeben werden.

4.52.3.13 void CSchnittstellenTest::fahrplanerstellungMenueaktivierung ()

Methode wird zum Aktivieren/Deaktivieren von Menueintraegen benoetigt

4.52.3.14 h_array< int,int >* CSchnittstellenTest::fahrplanerstellungMindesthaltezeit ()

Diese Methode uebergibt die Mindesthaltezeit fuer jede Haltestelle Die Daten werden in einem Hash_Array zurueckgegeben, indem der erste Wert die HaltestellenID und der zweite Wert die Haltezeit an dieser Haltestelle zurueckgegeben werden.

4.52.3.15 h_array< int,list< three_tuple< int,int,int >* >*>* CSchnittstellenTest::fahrplanerstellungMindestumsteigezeit ()

Diese Methode uebergibt die Mindestumsteigezeit fuer jede Kombination von Haltestelle und Liniennummern. Der Rueckgabewert baut sich wie folgt auf: Das erste Element im Hash_Array ist die HaltestellenID bei dem zweiten Element handelt es sich um eine Liste bestehend aus Liniennummer1, Liniennummer2 und die Mindestumsteigezeit zwischen diesen Linien.

4.52.3.16 void CSchnittstellenTest::fahrplanerstellungSukzessive-Loesungsverbesserung (int *zusammenhangskomponentenID*)

Starten der Berechnung der Lokalen Suche mit der entsprechenden Zusammenhangskomponete gekennzeichnet durch die ID

4.52.3.17 void CSchnittstellenTest::fahrplanerstellungWartezeiten (double *summeNetzbedingteWartezeiten*, double *wartezeitAufwand*, double *mindestWartezeitAufwand*, double *gesamtWarteaufwand*)

Uebergabe der Ergebnisse der Fahrplanberechnung hinsichtlich der Wartezeiten 1. Summe der netzbedingten Wartezeiten 2. Wartezeit-Aufwand 3. Mindestwartezeit-Aufwand 4. Gesamt-Wartezeit-Aufwand

4.52.3.18 void CSchnittstellenTest::fahrplanerstellungWegenetzplan ()

Diese Methode wird nur fuer Testzwecke benoetigt. Sie erstellt ein Netz der Transportverbindungen und ruft mit diesem Graphen die Klasse **CFahrplan** (S. 55) auf.

4.52.3.19 void CSchnittstellenTest::fahrplanerstellung-Zusammenhangskomponenten (array< list< three_tuple< int,int,int >* >* >* *zusammenhangskomponenten*)

Diese Methode erhaelt die Zusammenhangskomponente Als Parameter wird ein Array der Zusammenhangskomponenten uebergeben, in dem wiederum eine Liste von three_tuple steht Die Werte sind dabei 1. HaltestellenID, 2. Haltestellen-ID und LinienID

4.52.3.20 CGuetemasse * CSchnittstellenTest::holenGuetemasse ()

4.52.3.21 void CSchnittstellenTest::speichernGuetemasse (CGuetemasse * *guetemasse*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSchnittstellenTest.h**
- **CSchnittstellenTest.cpp**

4.53 CSimplexTableau Klassenreferenz

`#include <CSimplexTableau.h>`

Öffentliche Datenelemente

- **CSimplexTableau** (int, int, int)

- ~**CSimplexTableau** ()
- void **setzenStoppenFlag** ()
- void **einfuegenKantenInSimplextableau** (GRAPH<**CKnoten-Beschriftung***, **CKantenBeschriftung***>&)
- void **AusgebenSimplexTableau** ()
- bool **SuchenVerbesserung** (int, double &)
- void **startenLokaleSuche** (GRAPH<**CKnotenBeschriftung***, **CKantenBeschriftung***> &)

4.53.1 Beschreibung der Konstruktoren und Destruktoren

4.53.1.1 CSimplexTableau::CSimplexTableau (int *zeilen*, int *spalten*, int *taktzeit*)

Im Konstruktor werden die Anzahl Zeilen und Spalten und die Taktzeit uebergeben. Diese Werte werden in den entsprechenden Attributen gespeichert.

4.53.1.2 CSimplexTableau::~CSimplexTableau ()

Deklaration des Destruktors

4.53.2 Dokumentation der Elementfunktionen

4.53.2.1 void CSimplexTableau::AusgebenSimplexTableau ()

Hilfsfunktion zur Ausgabe des Simplextableau auf dem Bildschirm

4.53.2.2 bool CSimplexTableau::SuchenVerbesserung (int *zeile*, double & *reisezeitAufwand*)

Diese Methode sucht in der erhaltenen Zeile nach der Loesungsverbesserung und liefert bei Auffinden dieser true zurueck und ansonsten einen False-Wert Weiterhin wird bei Auffinden einer Verbesserung der aktuelle Reisezeitaufwand geaendert

4.53.2.3 void CSimplexTableau::einfuegenKantenInSimplextableau (GRAPH< CKnotenBeschriftung *,CKantenBeschriftung *>& *graph*)

Diese Funktion fuegt die Kanten in das Simplextableau ein. Dabei gehen wir wie folgt vor: Wir lesen die KantenID der Kante und merken sie uns kurz. Dann schauen wir nach, ob imMaximalgeruest in **CKantenBeschriftung** (S. 100) TRUE ist, ist dies der Fall, so fuegen wir sie in die naechste freie Spalte im

spaltenMapping ein. Diesen Spaltenwert schreiben wir in die Variable positionImSimplextableau in **CKantenBeschriftung** (S. 100). Analog mit den Nichtgeruestkanten hinsichtlich des zeilenMappings. Dann wird die Methode berechnenZyklusKanten fuer jede Nichtgeruestkante aufgerufen. Dann benutzen wir die Methode fuellenSimplexTableau, die den jeweiligen Stack und die Nichtgeruestkante bekommt, und die entsprechende Zeile im Simplextableau fuellt. Dann wird die Methode berechnenNetzbedingteWartezeit mit der Nichtgeruestkante und dem entsprechenden Stack aufgerufen, die den Loesungsvektor berechnet.

4.53.2.4 void CSimplexTableau::setzenStoppenFlag ()

setzen des Stoppen-Flag

4.53.2.5 void CSimplexTableau::startenLokaleSuche (GRAPH< CKnotenBeschriftung *,CKantenBeschriftung *>& *graph*)

Anstoss der Berechnung der Lokalen Suche auf dem existierenden Simplextableau Vorgehensweise: Anfangs wird die lokale Variable lokalOptimal auf false gesetzt. Dann wird solange iteriert bis die Bedingung lokalOptimal erfuellt ist. Im Koerper dieser Iteration wird eine Liste von Tupeln erstellt. Diese Tupeln enthalten zwei Werte: Zeilennummer und Reiseaufwand. Der Reiseaufwand ergibt sich aus netzbedingter Wartezeit multipliziert mit der Reisendenzahl. Diese Liste wird nach dem Reiseaufwand absteigend sortiert. Dann wird solange die Liste nicht leer ist und keine lokale Verbesserung gefunden worden ist iteriert. In diesem Koerper wird das aktuelle Listenelement genommen und fuer die entsprechende Zeile versucht, eine Verbesserung zu erzielen. Falls dieses moeglich wird die das Simplextableau und der Resultatsvektor geaendert und die lokale Variable gefundenVerbesserung auf true gesetzt, so dass dieser innere Koerper verlassen und die aeussere Ueberpruefung auf Lokale Optimalitaet, die nicht gegeben ist, ueberprueft. Falls keine Verbesserung erzielt werden kann, wird das naechste Listenelement betrachtet. Falls die Liste leer sein sollte und keine Verbesserung erreicht wurde, ist die gefunde Loesung lokal optimal.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSimplexTableau.h**
- **CSimplexTableau.cpp**

4.54 CStartverknuepfung Klassenreferenz

`#include <CStartverknuepfung.h>`

Öffentliche Datenelemente

- **CStartverknuepfung** (**CNetzplan*** meinNetzplan)
- **~CStartverknuepfung** ()
- void **waehlenBasislinie** (int algo)
- void **bildenEindeutigeLinienverknuepfung** ()
- void **erstellenLinien** ()

4.54.1 Ausführliche Beschreibung

Diese Klasse erzeugt aus einem Wegenetzgraphen anhand der Verknuepfungen "Basislinienwahl" und "eindeutige Linienwahl" einen groben Linienplan, der durch **CAbschlussverknuepfung** (S. 15) weiterbearbeitet wird.

Siehe Sonntag 108

Wenn ein Linienplan vorgegeben wurde, wird er in die interne Liste der Linien (CLinienliste) eingefuegt.

4.54.2 Beschreibung der Konstruktoren und Destruktoren

4.54.2.1 CStartverknuepfung::CStartverknuepfung (CNetzplan * *meinNetzplan*)

Der Konstruktor der Klasse. Er wird beim generieren der Klasse aufgerufen und sorgt fuer die Initialisierung der Klasse.

Eingabe:

- Ein Zeiger auf ein Objekt der Klasse **CNetzplan** (S. 121). Dieses Objekt enthaelt die fuer die Linienplanung noetigen Daten bereit.

Rueckgabe:

- nichts

4.54.2.2 CStartverknuepfung::~CStartverknuepfung ()

Der Destruktor wird aufgerufen, wenn die Klasse geloescht werden soll. In CStartverknuepfung werden die Basislinien wieder geloescht.

4.54.3 Dokumentation der Elementfunktionen

4.54.3.1 void CStartverknuepfung::bildenEindeutige-Linienverknuepfung ()

Aus der Menge der Basislinen sollen diejenigen miteinander verknuepft werden, die eine gemeinsame Endverknuepfung mit einfachem Belegungsgrad besitzen.

Weitere Infos siehe Flussdiagramm Sonntag Seite 114 und Text Sonntag ab Seite 113.

4.54.3.2 void CStartverknuepfung::erstellenLinien ()

Wenn die Linienplanerstellung mit Pflichtlinien aufgerufen wird, werden diese in die interenen Strukturen des Paketes Linienplan eingefuegt.

4.54.3.3 void CStartverknuepfung::waehlenBasislinie (int *algo*)

Basislinien sind Grundelemente des Verknuepfungsansatzes von Sonntag (siehe **CBasislinie** (S. 20)). Die Basislinienwahl erzeugt nun Basislinien, so dass jede Kante des Graphen mindestens einmal enthalten ist, die maximale Gesamtlinienlaenge nicht ueberschritten wird und Basislinien mit hohem Strom bevorzugt werden.

Weitere Infos siehe Flussdiagramm Sonntag Seite 112 und Text Sonntag ab Seite 108.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStartverknuepfung.h**
- **CStartverknuepfung.cpp**

4.55 CStrassennetz Klassenreferenz

`#include <CStrassennetz.h>`

Klassendiagramm für CStrassennetz:

Öffentliche Datenelemente

- **CStrassennetz** ()
- **CStrassennetz** (bool * abbruch, **CHSSchnittstelle** * **lnkSchnittstelle**, **CParameterHaltestelle** * **Parameter**, **CFortschritt** * **Fortschritt** = NULL)
- virtual ~**CStrassennetz** ()
- virtual void **starten** (void)

Geschützte Datenelemente

- void **initialisieren** (void)
- virtual void **berechnenHaltestellen** ()
- virtual void **erzeugenVerkehrsbedarfsmatrix** ()
- double **berechnenEntfernungen** (**CPunkt** * startpunkt, **CPunkt** * zielpunkt)
- list<**CPunkt** *>* **ermittelnKuerzesteWege** (double radius, **CPunkt** * startHaltestelle, **CPunkt** * zielHaltestelle)
- double **umrechnenMeterGRASS** (double meter)
- double **umrechnenGRASSMeter** (double grass)
- double **ermittelnEntfernung** (**CPunkt** * startpunkt, **CPunkt** * zielpunkt)

- double **ermittelnLaenge** (list<**CPunkt***> * Weg, unsigned int bisZuPosition = 0)
- **CPunkt*** **faellenLot** (**CPunkt** * haltestelle, bool NotfallsEndpunkteNehmen = false)
- **CPunkt*** **faellenLot** (**CPunkt** * haltestelle, **CPunkt** * linienanfang, **CPunkt** * linienende)
- **CPunkt*** **anpassenHaltestellenStrassennetz** (**CPunkt** * haltestelle)
- list<**CWegenetzKnoten***>* **anpassenHaltestellenStrassennetz** (list<**CWegenetzKnoten** *> * haltestellen)
- void **anpassenHaltestellenStrassennetz** ()
- void **bearbeitenUeberfluessigerHaltestellen** (void)
- **CPunkt*** **berechnenSchnittpunkt** (double m1, double t1, double m2, double t2)
- **CPunkt*** **berechnenPunkt** (double m, double t, double x)
- void **erzeugenStrassennetzGraph** ()
- void **ausgebenStrassennetzGraph** ()
- double* **rasternBevoelkerung** (list<**CBevoelkerungsdichteElement** *> * Bevoelkerungsdichtenliste)
- double* **rasternFlaeche** (void)
- void **rasternFahrtwuensche** (void)
- GRAPH<**CWegenetzKnoten** *, **CWegenetzKante** * >* **berechnenWegenetzgraph** (GRAPH<rat_point,rat_segment> ***Strassennetz**, list<**CWegenetzKnoten***> *Haltestellen)
- void **berechnenWegenetzgraph** (void)
- double **bewerten** (list<**CWegenetzKnoten***> * **HaltestellenKnoten**)
- void **suchenBegrenzungen** (list<list<**CPunkt***>* > * Strassennetzliste)
- void **holeKnoten** (edge* AktuelleKante, **CPunkt*** Start, **CPunkt*** Ziel)
- list<**CPunkt** *>* **suchenVerbindung** (**CPunkt** * StartPunkt, **CPunkt** * EndPunkt, bool AufHaltestellenPruefen = true)
- list<**CPunkt** *>* **suchenVerbindung** (node * Ziel, node * Start, bool AufHaltestellenPruefen = true)
- void **ausgebenWegenetzGraph** (GRAPH<**CWegenetzKnoten** *, **CWegenetzKante** *> * **Wegenetzgraph**)
- void **erstellenZusammenhangWegenetzgraph** (void)
- void **nummerierenHaltestellen** (void)
- void **ermittelnInzidenteKnoten** (edge * Kante, node *& linkerKnoten, node *& rechterKnoten)
- void **ermittelnInzidenteKnoten** (edge * Kante, node * linkerKnoten, node * rechterKnoten, list<node *> * linkeKnotenfolge, list<node *> * rechteKnotenfolge)
- void **aufbauenSektoren** (void)
- int **pruefenSektor** (rat_point p)
- void **einfuegenStrassensegmentSektor** (edge Kante)
- list<edge>* **sammelnStrassenAusSektoren** (int **Sektor**)

- void **ausgebenHaltestellen** ()
- list<node*>* **einfuegenHaltestellenStrassennetz** (GRAPH<rat_point,rat_segment> * **Strassennetz**, list<**CWegenetzKnoten***> * Haltestellen)
- void **kuerzenKoordinaten** (GRAPH<**CWegenetzKnoten***,**CWegenetzKante***> * **Wegenetzgraph**)
- void **anfordernSektoren** (double Ausdehnung)
- void **berechnenRealwegumgebung** (GRAPH<rat_point,rat_segment> *, node *)
- double **testobKreuzung** (GRAPH<rat_point,rat_segment> *, node *, double)
- double **pruefeLaenge** (GRAPH<rat_point,rat_segment> *, node *, node *, double)
- void **sortierenVerbindungsListe** (void)
- void **kuerzenStrassennetz** (void)
- double **berechnenModifizierterKoeffizientHaltestellenLage** (double Anteil)

Geschützte Attribute

- **CParameterHaltestelle* Parameter**
- matrix* **Verkehrsbedarfsmatrix**
- GRAPH<**CWegenetzKnoten** *, **CWegenetzKante** * >* **Wegenetzgraph**
- list<**CWegenetzKnoten***>* **HaltestellenKnoten**
- list<node*>* **HaltestellenImStrassennetz**
- GRAPH<rat_point,rat_segment>* **Strassennetz**
- list<two_tuple<**CPunkt***,int> >* **Fahrtwunschliste**
- bool* **abbrechen**
- double **begrenzungLinks**
- double **begrenzungRechts**
- double **begrenzungOben**
- double **begrenzungUnten**
- **CHSSchnittstelle* lnkSchnittstelle**
- matrix* **Bevoelkerungsraster**
- matrix* **Flaechenraster**
- unsigned int **FahrtwunschrasterEinheiten**
- matrix* **Fahrtwunschraster**
- unsigned int **SektorAnzahl**
- list<edge>** **Sektor**
- edge_array<double>* **dijkstraKanten**
- **CVisualisierung* StrassenVisualisierung**
- **CVisualisierung* BevoelkerungsversorgungVisualisierung**
- **CVisualisierung* FlaechendeckungVisualisierung**
- **CVisualisierung* LegendeVisualisierung**
- long double **MaximumBevoelkerung**
- long double **MaximumFlaeche**

- **CFortschritt* Fortschritt**
- list<**CVerbindung** * >* **verbindungsListe**
- **CGuetemasse* Guetemasse**
- bool **UeberHaltestellenGelaufen**
- list<node*>* **BesuchteKnotenListe**
- unsigned int **MaximaleAnzahlHaltestellen**

Statische geschützte Attribute

- const double **UmrechnungMeterGRASS** = 1.0
- const double **EntfernungsFaktor** = 0.6
- const double **SektorAusdehnung** = 60

4.55.1 Ausführliche Beschreibung

Dies ist eine Musterklasse zur Implementierung diverser Algorithmen zur Berechnung von Haltestellen. Soll ein spezieller Algorithmus implementiert werden, so wird hierzu eine weitere Klasse von dieser abgeleitet. Somit enthaelt diese Klasse nur Methoden, welche allgemeiner Natur sind. In der Basisklasse werden hierbei nur die Pflichthaltestellen als Ergebnis der Berechnung zurueckgegeben. Sie wird im Allgemeinen, nicht instanziert. Hierfuer sind die von ihr abgeleiteten Klassen vorgesehen.

4.55.2 Beschreibung der Konstruktoren und Destruktoren

4.55.2.1 CStrassennetz::CStrassennetz ()

Konstruktor. Dieser wird im Allgemeinen nicht verwendet, da ansonsten keine Ueberpruefung eines Abbruch-Signales von aussen stattfindet. Kann dazu verwendet werden, die Berechnung der Haltestellen in einem Batch-Betrieb ohne graphische Ausgabe zu erzeugen.

Siehe auch:
initialisieren(void) (S. 165) , Datenstruktur.CParameterHaltestelle

4.55.2.2 CStrassennetz::CStrassennetz (bool * *abbruch*, CHS-Schnittstelle * *lnkSchnittstelle*, CParameterHaltestelle * *Parameter*, CFortschritt * *Fortschritt* = NULL)

Konstruktor. Dies ist der Standard-Konstruktor, mit welchem aus dem Modul Programm eine Instanz von CStrassennetz oder auch einer abgeleiteten Klasse gebildet wird. Ist fuer das initialisieren der relevanten Dtaenstrukturen zustaendig.

Parameter:
abbruch Der Zeiger auf das Flag, welches den Abbruch signalisiert.

lnkSchnittstelle Zeiger auf die Schnittstelle aus dem Programm-Modul.

Parameter Zeiger auf das Objekt, welches alle Eingabedaten kapselt.

Siehe auch:
initialisieren(void) (S. 165) , Datenstruktur.CParameterHaltestelle

4.55.2.3 CStrassennetz::~CStrassennetz () [virtual]

Destruktor.

4.55.3 Dokumentation der Elementfunktionen

4.55.3.1 void CStrassennetz::anfordernSektoren (double *Ausdehnung*) [protected]

Fordert entsprechend der Ausdehnung des Planquadrates eine bestimmte Anzahl an Sektoren an. Dabei wird eine Sektor mit 'Ausdehnung'^2 angenommen.

@postcondition Wenigstens ein Sektor wurde angefordert.

4.55.3.2 void CStrassennetz::anpassenHaltestellenStrassennetz () [protected]

Passt alle vorhandenen Haltestellen an das Strassennetz an. Ruft iterativ fuer jede Haltestelle **CStrassennetz::anpassenHaltestellenStrassennetz**(CPunkt * haltestelle) (S. 159) auf.

Zudem werden alle Haltestellen entfernt, welche nach dem Ummappen zu nah (Entfernung < minimalesEinzugsgebiet) liegen. Hierbei wird die zuerst aufgefundene Haltestelle beibehalten. @postconditions Haltestellen sind an das Strassennetz angepasst.

Rückgabe:
Liefert eine Liste mit den angepassten Haltestellen zurueck.

Siehe auch:
Haltestelle.CStrassennetz**anpassenHaltestellenStrassennetz**(CPunkt *) (S. 159)

4.55.3.3 list< CWegenetzKnoten *>* CStrassennetz::anpassenHaltestellenStrassennetz (list< CWegenetzKnoten *>* *haltestellen*) [protected]

Passt alle vorhandenen Haltestellen an das Strassennetz an. Ruft iterativ fuer jede Haltestelle **CStrassennetz::anpassenHaltestellenStrassennetz**(CPunkt * haltestelle) (S. 159) auf. @postconditions Haltestellen sind an das Strassennetz angepasst.

Rückgabe:
Liefert eine Liste mit den angepassten Haltestellen zurueck.

Siehe auch:

Haltestelle.CStrassennetz**anpassenHaltestellenStrassennetz**(CPunkt *) (S. 159)

4.55.3.4 CPunkt * CStrassennetz::anpassenHaltestellenStrassennetz (CPunkt * *haltestelle*) [protected]

Pass eine gegebene Haltestelle an das Strassennetz an. Moegliche Implemntierungen sind LH 500, LH 510 und LH 520 bzw. Aufruf von **faellenLot**() (S. 164), usw. @postconditions Haltestelle ist an das Strassennetz angepasst.

Parameter:

haltestelle Haltestelle, welche an das Strassennetz angepasst werden soll.

4.55.3.5 void CStrassennetz::aufbauenSektoren (void) [protected]

Baut die Sektoren auf, welche die Kanten des Strassennetzes enthalten sollen. Die Idee eines Sektors ist dabei einem Quadtree nicht unaehnlich: Man teilt das gesamte Planquadrat in mehrere Einehiten, welche quadratisch angeordnet sind. Dadurch laesst sich fuer jeden Punkt eindeutig entscheiden, in welchem Sektor er liegt.

4.55.3.6 void CStrassennetz::ausgebenHaltestellen () [protected]

Gibt die Liste mit den Haltestellen aus.

4.55.3.7 void CStrassennetz::ausgebenStrassennetzGraph (void) [protected]

Erzeugen einer Ausgabe des StrassennetzGraphen

4.55.3.8 void CStrassennetz::ausgebenWegenetzGraph (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *Wegenetzgraph*) [protected]

Gibt den WegenetzGraph aus.

@preconditions Wird nur ausgeführt, wenn DEBUG definiert ist.

4.55.3.9 void CStrassennetz::bearbeitenUeberfluessigerHaltestellen (void) [protected]

4.55.3.10 double CStrassennetz::berechnenEntfernungen (CPunkt * *startpunkt*, CPunkt * *zielpunkt*) [protected]

Berechnet die Entfernung zwischen zwei Punkten

- arbeitet auf dem Strassennetzgraph
- berechnet die Entfernung fuer den ganzen Graphen, fuer die 2 uebergebenen Punkte, die auf der Strasse liegen muessen @preconditions zwei Punkte im Strassennetz mit Kreuzungen die zwei uebergebenen Punkte muessen auf der Strasse liegen Kreuzungen muessen berechnet worden sein @postconditions Entfernung zwischen den beiden Punkten

Parameter:

startpunkt Erster Punkt, von dem aus die Entfernung gemessen wird.

zielpunkt Zweiter Punkt, zu dem die Entfernung gemessen wird.

4.55.3.11 void CStrassennetz::berechnenHaltestellen () [protected, virtual]

Berechnet die Haltestellen. Diese Methode wird von den abgeleiteten Klassen ueberschrieben.

Erneute Implementation in **CStrassennetzES** (S. 173), **CStrassennetzStochastisch** (S. 176), und **CStrassennetzStochastischUmgebung** (S. 177).

4.55.3.12 double CStrassennetz::berechnenModifizierterKoeffizientHaltestellenLage (double *Anteil*) [protected]

Dies ist ein Koeffizient, welcher dazu dient, die Anpassung der Haltestellen zusaetzlich von der Entfernung der zuerst berechneten Haltestellenlage

4.55.3.13 CPunkt * CStrassennetz::berechnenPunkt (double *m*, double *t*, double *x*) [protected]

Hilfsfunktion, die einen Punkt einer Geraden berechnet und zurückgibt

Parameter:

m Steigungskoeffizient.

t Y-Versatz.

x Wert.

4.55.3.14 void CStrassennetz::berechnenRealwegumgebung (GRAPH< rat_point,rat_segment >* *G*, node * *Haltestelle*) [protected]

Zur Realwegradiusberechnung um eine Haltestelle herum

4.55.3.15 CPunkt * CStrassennetz::berechnenSchnittpunkt (double *m1*, double *t1*, double *m2*, double *t2*) [protected]

Hilfsfunktion, die den Schnittpunkt zweier Geraden berechnet und zurückgibt

Parameter:

m1 Steigungskoeffizient der ersten Geraden.

t1 Y-Versatz der ersten Geraden.

m2 Steigungskoeffizient der zweiten Geraden.

t2 Y-Versatz der zweiten Geraden.

4.55.3.16 void CStrassennetz::berechnenWegenetzgraph (void) [protected]

Ruft die parametrisierte Version von **berechnenWegenetzgraph**() (S. 161) auf und stellt das Ergebnis in Wegenetzgraph zur Verfuegung.

4.55.3.17 GRAPH< CWegenetzKnoten *,CWegenetzKante * >* CStrassennetz::berechnenWegenetzgraph (GRAPH< rat_point,rat_segment >* *Strassennetz*, list< CWegenetzKnoten *>* *Haltestellen*) [protected]

Berechnet dem Reza sein Wegenetzgraph. Dabei werden zunaechst aus dem Strassennetz alle Kreuzungen und Start- und Endpunkte von Strassenzuegen entfernt. Somit bleiben zum Schluss nur noch Haltestellen als Knoten im Graphen uebrig. Beim entfernen von Kreuzungen werden Kanten doppelt auftreten. Sobald eine Kante zwischen zwei Knoten eingefuegt werden soll, wo bereits eine Kante existiert, wird ihre Gesamtlaenge entlang der realen Strassen (!) verglichen und nur die kuerzere Kante gespeichert, die andere wird verworfen (geloescht). @precondition Strassennetz existiert komplett, Haltestellen sind gesetzt.

4.55.3.18 double CStrassennetz::bewerten (list< CWegenetzKnoten *>* *HaltestellenKnoten*) [protected]

Gibt eine Berwertung der uebergebenen Haltestellen zurueck. Dabei werden beruecksichtigt: (Bevoelkerungsabdeckung) (Flaechenabdeckung) (Fahrtwunschabdeckung) Doppelte Abdeckungen werden als Strafe hinzugezaehlt.

4.55.3.19 list< node *>* CStrassennetz::einfuegenHaltestellen-Strassennetz (GRAPH< rat_point,rat_segment >* *Strassennetz*, list< CWegenetzKnoten *>* *Haltestellen*) [protected]

Fuegt die Haltestellen in das Strassennetz ein. Dient als Grundlage fuer die Erstellung des Wegenetzgraphen. Dabei werden die Haltestellen als Knoten mit negativen Vorzeichen behandelt, um sie spaeter wieder erkennen zu koennen.

Parameter:

Strassennetz Das Strassennetz, in welches die Haltestellen eingefuegt werden sollen.

Haltestellen Die Haltestellen, welche eingefuegt werden sollen.

Rückgabe:
Die Liste der neu erzeugten Knoten, in der gleichen Reihenfolge, wie die zugehoerigen Haltestellen.

4.55.3.20 void CStrassennetz::einfuegenStrassensegmentSektor (edge *Kante*) [protected]

Fuegt eine Linie in die beruehrten Sektoren ein. Verwendet **pruefenSektor**(rat_point) (S. 165).

4.55.3.21 double CStrassennetz::ermittelnEntfernung (CPunkt * *startpunkt*, CPunkt * *zielpunkt*) [protected]

4.55.3.22 void CStrassennetz::ermittelnInzidenteKnoten (edge * *Kante*, node * *linkerKnoten*, node * *rechterKnoten*, list< node *>* *linkeKnotenfolge*, list< node *>* *rechteKnotenfolge*) [protected]

Ermittelt die Knoten rechts und links von einer Kante, die Kreuzungen darstellen. Knoten, die keine Kreuzungen sind werden in eine Liste eingetragen und zurueckgegeben

Parameter:
Kante @preconditions Alle Knoten haben das Attribut Haltestelle, Kreuzung, Strassenpunkt

4.55.3.23 void CStrassennetz::ermittelnInzidenteKnoten (edge * *Kante*, node *& *linkerKnoten*, node *& *rechterKnoten*) [protected]

Ermittelt die Knoten rechts und links von einer Kante, die Kreuzungen darstellen. Knoten, die keine Kreuzungen sind werden ignoriert

Parameter:
Kante @preconditions Alle Knoten haben das Attribut Haltestelle, Kreuzung, Strassenpunkt

4.55.3.24 list< CPunkt *>* CStrassennetz::ermittelnKuerzesteWege (double *radius*, CPunkt * *startHaltestelle*, CPunkt * *zielHaltestelle*) [protected]

Ermittelt den kuerzesten Weg zwischen zwei Haltestellen

- arbeitet auf dem Strassennetzgraphen
- berechnet die kuerzesten Wege zwischen zwei Haltestellen mit Hilfe von LEDA-Algorithmen Implementiert LH 900 bzw. Aufruf einer gleichwertigen Methode in LEDA. @preconditions Strassennetz mit Kreuzungen wurde berechnet Die zwei uebergebenen Punkte im Strassennetz auf Strassen @postconditions Entfernung

Parameter:

radius Die Entfernung, in welcher die Wege zuruekcgegeben werden sollen.

startHaltestelle Haltestelle, von der aus der Weg gesucht werden soll.

zielHaltestelle Haltestelle, zu welcher der Weg gesucht werden soll.

4.55.3.25 double CStrassennetz::ermittelnLaenge (list< CPunkt *>* *Weg*, unsigned int *bisZuPosition* = 0) [protected]

4.55.3.26 void CStrassennetz::erstellenZusammenhang-Wegenetzgraph (void) [protected]

Erstellt aus dem unter Umstaenden noch unzusammenhaengenden Wegenetz-Graphen einen zusammenhaengenden. Hierzu wird eine Variante von Kruskal vewendet, wobei davon ausgegangen wird, dass bis zum Verbindungsradius schon entsprechend vorgearbeitet wurde und nur noch Verbindungen groesser als der Verbindungsradius geprueft werden muessen. Als Vorgabe dient dabei die verbindungsListe.

Siehe auch:

sortierenVerbindungsListe(void) (S. 166) , **berechnen-Wegenetzgraph**(void) (S. 161)

4.55.3.27 void CStrassennetz::erzeugenStrassennetzGraph () [protected]

Erstellt den StrassennetzGraph. Ruft hierzu die Methode LEDA::SWEEP_-SEGMENTS() auf.

4.55.3.28 void CStrassennetz::erzeugenVerkehrsbedarfsmatrix () [protected, virtual]

Erzeugt die Verkehrsbedarfsmatrix, die an die Schnittstelle zurückgegeben wird. verwendet LH 540.

4.55.3.29 CPunkt * CStrassennetz::faellenLot (CPunkt * *haltestelle*, CPunkt * *linienanfang*, CPunkt * *linienende*) [protected]

Faellt das Lot von der angegebenen Haltestelle auf die gegebene Linie.

Rückgabe:

Der **CPunkt** (S. 140), welcher den Lotfusspunkt beschreibt oder NULL, falls dieser auf dieser nicht auf der Linie liegt.

4.55.3.30 CPunkt * CStrassennetz::faellenLot (CPunkt * *haltestelle*, bool *NotfallsEndpunkteNehmen* = false) [protected]

Legt die Haltestelle an den naechsten Strassenzug. Hierzu wird das Lot benutzt. Implmetiert LH 500. @preconditions Haltestelle darf auch auf dem Strassennetz liegen! @postconditions Haltestelle wurd am Strassenentz liegend auf den naechsten Strassenzug gelegt.

Parameter:

haltestelle Haltestelle, von der aus auf den naechsten Strassenzug das Lot gefaellt werden soll.

4.55.3.31 void CStrassennetz::holeKnoten (edge * *AktuelleKante*, CPunkt * *Start*, CPunkt * *Ziel*) [protected]

Gibt die zugehoerigen Knoten zur uebergebenen Kante zurueck.

4.55.3.32 void CStrassennetz::initialisieren (void) [protected]

Dient dem Initialiseren aller Variablen und Zeiger. Wird direkt von den Konstruktoren aufgerufen, welche danach weitere Modifikationen am Zustand vornehmen. Ruft CHSSchnittstelle::haltestellenWerteanforderung auf.

Siehe auch:

Programm.CHSSchnittstellehaltestellenWerteanforderung(float &,float &,float &,float &,list *,list*> *,list *,matrix *,list *,matrix *,list *)

4.55.3.33 void CStrassennetz::kuerzenKoordinaten (GRAPH< CWegenetzKnoten *,CWegenetzKante *>* *Wegenetzgraph*) [protected]

Kuerzt die Koordinaten der Knoten (=Haltestellen) im uebergebenen Graphen auf zwei Nachkommastellen.

4.55.3.34 void CStrassennetz::kuerzenStrassennetz (void) [protected]

Entfernt aus dem Strassennetz lose Teile. Hierzu werden alle Komponenten berechnet und dann alle Teile, welche nicht zur groessten gehoeren entfernt.

4.55.3.35 void CStrassennetz::nummerierenHaltestellen (void) [protected]

Nummeriert die vorhandenen Haltestellen mit Null beginnend durch.

@postonditions Alle Haltestellen aus HaltestellenKnoten sind mit 0 beginnend fortlaufend durchnummeriert.

4.55.3.36 double CStrassennetz::pruefeLaenge (GRAPH< rat_point,rat_segment >* *G*, node * *Knoten1*, node * *Knoten2*, double *aktuelleLaenge*) [protected]

4.55.3.37 int CStrassennetz::pruefenSektor (rat_point *p*) [protected]

Prueft den Sektor, in welchem sich ein Punkt befindet.

4.55.3.38 double * CStrassennetz::rasternBevoelkerung (list< CBevoelkerungsdichteElement *>* *Bevoelkerungsdichtenliste*) [protected]

Rastert die vorhandene Bevoelkerung unter Beruecksichtigung von Bevoelkerungsraster und BevoelkerungsrasterEinheiten. @precondition Bevoelkerungsraster ist initialisert und erzeugt.

Siehe auch:
Bevoelkerungsraster (S. 168) , BevoelkerungsrasterEinheiten.

4.55.3.39 void CStrassennetz::rasternFahrtwuensche (void) [protected]

Rastert die vorhandenen Fahrtwuenche unter Beruecksichtigung von Fahrtwunschsraster und FahrtwunschrasterEinheiten. @precondition Fahrtwunschraster ist initialisert und erzeugt.

Siehe auch:
Fahrtwunschraster (S. 169) , **FahrtwunschrasterEinheiten** (S. 169).

4.55.3.40 double * CStrassennetz::rasternFlaeche (void) [protected]

Rastert die vorhandene Flaeche unter Beruecksichtigung von Bevoelkerungsraster und BevoelkerungsrasterEinheiten. @precondition Bevoelkerungsraster ist initialisert und erzeugt.

Siehe auch:
Flaechenraster (S. 169) , BevoelkerungsrasterEinheiten.

4.55.3.41 list< edge >* CStrassennetz::sammelnStrassenAusSektoren (int *Sektor*) [protected]

Liefert alle Strassen aus dem uebergebenen Sektor plus den direkt angrenzenden Sektoren. Dabei werden niemals Kanten doppelt ausgegeben.

Parameter:
Sektor Sektor, welcher der Ursprung der Ausgabe ist.

Rückgabe:
Eine Liste alle Kanten, welche in dem Sektor und aus den umliegenden ermittelt wurden, ohne doppelte Kanten.

4.55.3.42 void CStrassennetz::sortierenVerbindungsListe (void) [protected]

Sortiert die verbindungsListe, bestehend aus Zeigern auf Objekten des Typs **CVerbindung** (S. 207), aufsteigend nach ihrer Laenge. Wird zur Vorbereitung von **erstellenZusammenhangWegenetzgraph**() (S. 164) benoetigt.

Siehe auch:
erstellenZusammenhangWegenetzgraph(void) (S. 164)

4.55.3.43 void CStrassennetz::starten (void) [virtual]

Führt die Berechnungen aus. Kann in den abgeleiteten Klassen erweitert werden. Ansonsten werden die folgenden Schritte ausgeführt:

- Haltestellen setzen,
- Haltestellen an das Strassennetz anpassen,
- Verkehrsbedarfsmatrix erzugen,
- Wegenetzgraph erstellen,
- aufrufen von **CHSSchnittstelle::haltestellenErgebnisrueckgabe** (S. 99) .

Siehe auch:
Programm.CHSSchnittstellehaltestellenErgebnisrueckgabe(GRAPH *,matrix *)

Erneute Implementation in **CStrassennetzES** (S. 174), und **CTest-Strassennetz** (S. 200).

4.55.3.44 void CStrassennetz::suchenBegrenzungen (list< list< CPunkt *>* >* *Strassennetzliste*) [protected]

Bestimmt die Ausdehnung des uebergebenen Strassennetzes und traegt die vorgefundenen Werte als Begrenzungen ein.

Parameter:
Strassennetzliste Eine Liste von Listen aus Punkten, welche das Strassennetz repraesentiert.

4.55.3.45 list< CPunkt *>* CStrassennetz::suchenVerbindung (node * *Ziel*, node * *Start*, bool *AufHaltestellenPruefen* = true) [protected]

Sucht auf dem strassennetz die kuerzeste Verbindung vom StartPunkt zum EndPunkt. Wird dabei ueber eine Haltestelle gelaufen, so wird kein Pfad zurueckgeben, ansonsten enthaelt der Graph eine Liste der Punkte, welchen den abzufahrenden Linienzug einschliesslich StartPunkt und EndPunkt, enthaelt.

Parameter:

StartPunkt Punkt, von welchem die Verbindung aufgebaut werden soll.

EndPunkt Punkt, zu welchem die Verbindung aufgebaut werden soll.

AufHaltestellenPruefen Gibt an, ob auf Haltestellen geprueft werden soll, also ob ein Pfad auch dann gueltig ist, wenn er ueber eine Haltestelle laeuft.

Rückgabe:

Liste mit dem Linienzug oder NULL.

4.55.3.46 list< CPunkt *>* CStrassennetz::suchenVerbindung (CPunkt * *StartPunkt*, CPunkt * *EndPunkt*, bool *AufHaltestellenPruefen* = true) [protected]

Sucht auf dem strassennetz die kuerzeste Verbindung vom StartPunkt zum EndPunkt. Wird dabei ueber eine Haltestelle gelaufen, so wird kein Pfad zurueckgeben, ansonsten enthaelt der Graph eine Liste der Punkte, welchen den abzufahrenden Linienzug einschliesslich StartPunkt und EndPunkt, enthaelt.

Parameter:

StartPunkt Punkt, von welchem die Verbindung aufgebaut werden soll.

EndPunkt Punkt, zu welchem die Verbindung aufgebaut werden soll.

AufHaltestellenPruefen Gibt an, ob auf Haltestellen geprueft werden soll, also ob ein Pfad auch dann gueltig ist, wenn er ueber eine Haltestelle laeuft.

Rückgabe:

Liste mit dem Linienzug oder NULL.

4.55.3.47 double CStrassennetz::testobKreuzung (GRAPH< rat_point,rat_segment >* *G*, node * *dieserKnoten*, double *aktuelleLaenge*) [protected]

4.55.3.48 double CStrassennetz::umrechnenGRASSMeter (double *grass*) [protected]

4.55.3.49 double CStrassennetz::umrechnenMeterGRASS (double *meter*) [protected]

4.55.4 Dokumentation der Datenelemente

4.55.4.1 list< node *>* CStrassennetz::BesuchteKnotenListe [protected]

Liste der besuchten Knoten bei der Realwegumgebung einer Haltestelle

4.55.4.2 matrix * CStrassennetz::Bevoelkerungsraster [protected]

Enthaelt das Bevoelkerungsaufkommen, wobei selbiges ueber das Planquadrat gerastert ist. Die Rasterung wird duch BevoelkerungsrasterEinheiten vorgegeben.

4.55.4.3 CVisualisierung * CStrassennetz::Bevoelkerungsversorgung-Visualisierung [protected]

Eine spezielle Klasse, welche das Visualisieren der Bevoelkerungsversorgung erlaubt.

4.55.4.4 const double CStrassennetz::EntfernungsFaktor = 0.6 [static, protected]

Dient der Umrechnung der Entfernung (Luftlinie) zur Annaeherung an die Entfernung unter Beruecksichtigung des Realweges.

Entspringt einem Erfahrungswert der HCR.

4.55.4.5 list< two_tuple< CPunkt *,int > >* CStrassennetz::Fahrtwunschliste [protected]

Das Attribut Fahrtwunschliste dient dazu die Eintraege der Fahrtwunschmatrix einem Punkt zuzuordnen.

4.55.4.6 matrix * CStrassennetz::Fahrtwunschraster [protected]

Enthaelt die Fahrtwuensche, wobei selbige ueber das Planquadrat gerastert ist. Die Rasterung wird duch FahrtwunschrasterEinheiten vorgegeben.

4.55.4.7 unsigned int CStrassennetz::FahrtwunschrasterEinheiten [protected]

Beschreibt die Anzhal der Rasterungseinheiten fuer die Fahrtwuensche.

4.55.4.8 CVisualisierung * CStrassennetz::Flaechendeckung-Visualisierung [protected]

Eine spezielle Klasse, welche das Visualisieren der Flaechenversorgung erlaubt.

4.55.4.9 matrix * CStrassennetz::Flaechenraster [protected]

Enthaelt die Flaechenversorgung, wobei selbiges ueber das Planquadrat gerastert ist. Die Rasterung wird duch BevoelkerungsrasterEinheiten vorgegeben.

4.55.4.10 CFortschritt * CStrassennetz::Fortschritt [protected]

Eine Instanz, welche das graphische Ausgeben von Informationen zum Fortschritt der Berechnung erlaubt.

4.55.4.11 CGuetemasse * CStrassennetz::Guetemasse [protected]

Zeiger auf die Guetemasse, welche an die Schnitstelle zurueckgeliefert werden. Dieses Objekt wird bei der Initialisierung erzeugt, aber nur durch die Schnittstelle vernichtet!

4.55.4.12 list< node *>* CStrassennetz::HaltestellenImStrassennetz [protected]

Beinhaltet die Liste der in das Strassennetz eingefuegten Haltestellen. Diese sind dabei als node vermerkt, um sie im Strassennetz wiederzufinden.

4.55.4.13 list< CWegenetzKnoten *>* CStrassennetz::Haltestellen-Knoten [protected]

Stellt einen Zeiger auf eine LEDA-Liste dar, die WegenetzKnoten als Elemente enthaelt. Dies sind die berechnenten Haltestellen, sowie die Pflichthaltestellen. Zudem sind ID und Prioritaet vergeben.

4.55.4.14 CVisualisierung * CStrassennetz::LegendeVisualisierung [protected]

Eine spezielle Klasse, welche das Visualisieren der Legende erlaubt.

4.55.4.15 unsigned int CStrassennetz::MaximaleAnzahlHaltestellen [protected]

Liste der Randknoten, die das Realwegumgebungspolygon der Haltestelle bilden

4.55.4.16 long double CStrassennetz::MaximumBevoelkerung [protected]

Enthaelt das Maximum aus der Bevoelkerungsrasterung.

4.55.4.17 long double CStrassennetz::MaximumFlaeche [protected]

Enthaelt das Maximum aus der Flaechenrasterung.

4.55.4.18 CParameterHaltestelle * CStrassennetz::Parameter [protected]

4.55.4.19 list< edge >** CStrassennetz::Sektor [protected]

Repraesentiert die Liste mit den Kanten in den Sektoren.

4.55.4.20 unsigned int CStrassennetz::SektorAnzahl [protected]

Beschreibt die Anzahl der Sektoren, welche ueber das Planquadrat verteilt werden. wird dynamisch zur Laufzeit unter Beruecksichtigung SektorAusdehnung ermittelt.

4.55.4.21 const double CStrassennetz::SektorAusdehnung = 60 [static, protected]

Gibt an, welche Ausdehnung ein Sektor hat. Diese Sektoren dienen dabei einem dem Hashing sehr aenlichen Verfahren.

4.55.4.22 CVisualisierung * CStrassennetz::StrassenVisualisierung [protected]

Eine spezielle Klasse, welche das Visualisieren des Strassennetzes und der Haltestellen erlaubt.

4.55.4.23 GRAPH< rat_point,rat_segment >* CStrassennetz::Strassennetz [protected]

Das Strassennetz ist ein Graph und stellt das eigentliche Strassennetz dar. Es wird durch die Methode berechnenKreuzungen() aufbebaut. Die Knoten stellen die Kreuzungen dar, die Kanten die Strassen zwischen den Kreuzungen

4.55.4.24 bool CStrassennetz::UeberHaltestellenGelaufen [protected]

Enthaelt ein Flag, ob **suchenVerbindung**() (S. 168) ueebr eine Haltestelle gelaufen ist, oder nicht.

4.55.4.25 const double CStrassennetz::UmrechnungMeterGRASS = 1.0 [static, protected]

Dient der Umrechnung von Meter nach GRASS. Die Zahl kennzeichnet dabei wieviel GRASS-Einheiten einem Meter entsprechen.

4.55.4.26 matrix * CStrassennetz::Verkehrsbedarfsmatrix [protected]

4.55.4.27 GRAPH< CWegenetzKnoten *,CWegenetzKante * >* CStrassennetz::Wegenetzgraph [protected]

Dem Reza sein Wegenetzgraph.

Die Knoten des Wegenetzgraphen stellen die Haltestellen dar. Sie haben als Attribute eine ID, eine Prioritaet, die im Bereich von 1 bis 10 liegt und die Wichtigkeit einer Haltestelle angeben, sowie eine Position im Koordinatensystem des GIS.

Die Kanten habe lediglich das Attribut Entfernung, das die Entfernung zweier Haltestellen in Metern angibt. Sie stellen die Verbindung zwischen den Haltestellen ohne festen Bezug zum GIS oder Strassennetz dar.

4.55.4.28 bool * CStrassennetz::abbrechen [protected]

Verweist auf das Attribut "abbrechen" der Klasse "CHaltestellte". Dient somit dem Pruefen auf einen Abbruchbefehl.

Siehe auch:
Haltestelle.CHaltestelle

4.55.4.29 double CStrassennetz::begrenzungLinks [protected]

Begrenzt das zu betrachtende Planquadrat nach links(=Westen).

4.55.4.30 double CStrassennetz::begrenzungOben [protected]

Begrenzt das zu betrachtende Planquadrat nach oben(=Norden).

4.55.4.31 double CStrassennetz::begrenzungRechts [protected]

Begrenzt das zu betrachtende Planquadrat nach rechts(=Osten).

4.55.4.32 double CStrassennetz::begrenzungUnten [protected]

Begrenzt das zu betrachtende Planquadrat nach unten(=Sueden).

4.55.4.33 edge_array< double >* CStrassennetz::dijkstraKanten [protected]

Die Kosten jeder Kante fuer den Dijkstra.

4.55.4.34 CHSSchnittstelle * CStrassennetz::lnkSchnittstelle [protected]

Verweist auf die Schnittstellenklasse. Wird von der aufrufenden Klasse uebergeben und auch von dieser vernichtet.

4.55.4.35 list< CVerbindung * >* CStrassennetz::verbindungsListe [protected]

Enthaelt die Liste mit den noch zu pruefenden Verbindungen, um den Wegenetzgraph zusammenhaengend zu machen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStrassennetz.h**
- **CStrassennetz.cpp**

4.56 CStrassennetzES Klassenreferenz

`#include <CStrassennetzES.h>`

Klassendiagramm für CStrassennetzES:

Öffentliche Datenelemente

- **CStrassennetzES** ()
- **CStrassennetzES** (bool * **abbrechen**, **CHSSchnittstelle** * **lnkSchnittstelle**, **CParameterHaltestelle** * **Parameter**, **CFortschritt** * **Fortschritt** = NULL)
- void **berechnenHaltestellen** ()
- void **starten** (void)
- void **setzenHaltestelleES** (**CPunkt** * DerPunkt)
- virtual teaValue* **getValuePrototype** ()
- virtual teaValue* **calcFitness** (teaIndividual *it, teaValue *itsValue)
- virtual teaValue* **getConstraintsValuePrototype** ()
- virtual teaValue* **checkConstraints** (teaIndividual *it, teaValue* itsValue)

4.56.1 Ausführliche Beschreibung

Dies ist eine Spezialisierung von **CStrassennetz** (S. 155). Hierin werden die Haltestellen mittels einer ES ueber das Planquadrat verteilt. Ansonsten wird der in **CStrassennetz** (S. 155) definierte Durchlauf unveraendert uebernommen.

Siehe auch:

Haltestelle.CStrassennetz , Haltestelle.CStrassennetz**starten**(void) (S. 174) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170)

4.56.2 Beschreibung der Konstruktoren und Destruktoren

4.56.2.1 CStrassennetzES::CStrassennetzES ()

4.56.2.2 CStrassennetzES::CStrassennetzES (bool * *abbrechen*, CHSSchnittstelle * *lnkSchnittstelle*, CParameterHaltestelle * *Parameter*, CFortschritt * *Fortschritt* = NULL)

4.56.3 Dokumentation der Elementfunktionen

4.56.3.1 void CStrassennetzES::berechnenHaltestellen () [virtual]

Berechnet die Haltestellen, wobei selbige zufaellig im Planquadrat verteilt werden. Verwendet setzenHaltestelleStochastisch(CPunkt * DerPunkt).

Siehe auch:

void setzenHaltestelleStochastisch(CPunkt * DerPunkt) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170) @postcondition Die Liste aus **HaltestellenKnoten** (S. 170) ist erzeugt.

Implementiert von **CStrassennetz** (S. 160).

4.56.3.2 teaValue * CStrassennetzES::calcFitness (teaIndividual * *it*, teaValue * *itsValue*) [virtual]

4.56.3.3 teaValue * CStrassennetzES::checkConstraints (teaIndividual * *it*, teaValue * *itsValue*) [virtual]

4.56.3.4 teaValue * CStrassennetzES::getConstraintsValuePrototype () [inline, virtual]

4.56.3.5 teaValue * CStrassennetzES::getValuePrototype () [inline, virtual]

4.56.3.6 void CStrassennetzES::setzenHaltestelleES (CPunkt * *DerPunkt*)

Ermittelt genau eine Haltestelle nach dem Zufallsprinzip.

4.56.3.7 void CStrassennetzES::starten (void) [virtual]

Führt die Berechnungen aus. Kann in den abgeleiteten Klassen erweitert werden. Ansonsten werden die folgenden Schritte ausgeführt:

- Haltestellen setzen,

- Haltestellen an das Strassennetz anpassen,
- Verkehrsbedarfsmatrix erzugen,
- Wegenetzgraph erstellen,
- aufrufen von **CHSSchnittstelle::haltestellenErgebnisrueckgabe** (S. 99) .

Siehe auch:

Programm.CHSSchnittstellehaltestellenErgebnisrueckgabe(GRAPH *,matrix *)

Implementiert von **CStrassennetz** (S. 167).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStrassennetzES.h**
- **CStrassennetzES.cpp**

4.57 CStrassennetzGraph Klassenreferenz

`#include <CStrassennetzGraph.h>`

Öffentliche Datenelemente

- void **erzeugen** (void)

4.57.1 Dokumentation der Elementfunktionen

4.57.1.1 void CStrassennetzGraph::erzeugen (void)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStrassennetzGraph.h**
- **CStrassennetzGraph.cpp**

4.58 CStrassennetzStochastisch Klassenreferenz

`#include <CStrassennetzStochastisch.h>`

Klassendiagramm für CStrassennetzStochastisch:

Öffentliche Datenelemente

- **CStrassennetzStochastisch** ()
- **CStrassennetzStochastisch** (bool * **abbrechen**, **CHSSchnittstelle** * **lnkSchnittstelle**, **CParameterHaltestelle** * **Parameter**, **CFortschritt** * **Fortschritt** = NULL)
- void **berechnenHaltestellen** ()
- void **setzenHaltestelleStochastisch** (**CPunkt** * DerPunkt)

4.58.1 Ausführliche Beschreibung

Dies ist eine Spezialisierung von **CStrassennetz** (S. 155). Hierin werden die Haltestellen nach dem Zufallsprinzip ueber das Planquadrat verteilt. Ansonsten wird der in **CStrassennetz** (S. 155) definierte Durchlauf unveraendert uebernommen.

Siehe auch:

Haltestelle.CStrassennetz , Haltestelle.CStrassennetz**starten**(void) (S. 167) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170)

4.58.2 Beschreibung der Konstruktoren und Destruktoren

4.58.2.1 CStrassennetzStochastisch::CStrassennetzStochastisch ()

4.58.2.2 CStrassennetzStochastisch::CStrassennetzStochastisch (bool * *abbrechen*, CHSSchnittstelle * *lnkSchnittstelle*, CParameterHaltestelle * *Parameter*, CFortschritt * *Fortschritt* = NULL)

4.58.3 Dokumentation der Elementfunktionen

4.58.3.1 void CStrassennetzStochastisch::berechnenHaltestellen () [virtual]

Berechnet die Haltestellen, wobei selbige zufaellig im Planquadrat verteilt werden. Verwendet **setzenHaltestelleStochastisch**(CPunkt * DerPunkt) (S. 176).

Siehe auch:

void **setzenHaltestelleStochastisch**(CPunkt * DerPunkt) (S. 176) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170) @postcondition Die Liste aus **HaltestellenKnoten** (S. 170) ist erzeugt.

Implementiert von **CStrassennetz** (S. 160).

4.58.3.2 void CStrassennetzStochastisch::setzenHaltestelleStochastisch (CPunkt * *DerPunkt*)

Ermittelt genau eine Haltestelle nach dem Zufallsprinzip.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStrassennetzStochastisch.h**
- **CStrassennetzStochastisch.cpp**

4.59 CStrassennetzStochastischUmgebung Klassenreferenz

`#include <CStrassennetzStochastischUmgebung.h>`

Klassendiagramm für CStrassennetzStochastischUmgebung:

Öffentliche Datenelemente

- **CStrassennetzStochastischUmgebung** ()
- **CStrassennetzStochastischUmgebung** (bool * **abbrechen**, **CHS-Schnittstelle** * **lnkSchnittstelle**, **CParameterHaltestelle** * **Parameter**, **CFortschritt** * **Fortschritt** = NULL)
- void **berechnenHaltestellen** ()
- void **setzenHaltestelleStochastisch** (**CPunkt** * DerPunkt)

4.59.1 Ausführliche Beschreibung

Dies ist eine Spezialisierung von **CStrassennetz** (S. 155). Hierin werden die Haltestellen nach dem Zufallsprinzip ueber das Planquadrat verteilt. Ist dabei in einer Umgebung schon eine Haltestelle vorhanden, so wird keine gesetzt. Ansonsten wird der in **CStrassennetz** (S. 155) definierte Durchlauf unveraendert uebernommen.

Siehe auch:

Haltestelle.CStrassennetz , Haltestelle.CStrassennetz**starten**(void) (S. 167) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170)

4.59.2 Beschreibung der Konstruktoren und Destruktoren

4.59.2.1 CStrassennetzStochastischUmgebung::CStrassennetz-StochastischUmgebung ()

4.59.2.2 CStrassennetzStochastischUmgebung::CStrassennetz-StochastischUmgebung (bool * *abbrechen*, CHSSchnittstelle * *lnkSchnittstelle*, CParameterHaltestelle * *Parameter*, CFortschritt * *Fortschritt* = NULL)

4.59.3 Dokumentation der Elementfunktionen

4.59.3.1 void CStrassennetzStochastischUmgebung::berechnen-Haltestellen () [virtual]

Berechnet die Haltestellen, wobei selbige zufaellig im Planquadrat verteilt werden. Ist dabei in einer Umgebung schon eine Haltestelle vorhanden, so wird keine gesetzt. Verwendet **setzenHaltestelleStochastisch**(CPunkt * DerPunkt) (S. 177).

Siehe auch:

void **setzenHaltestelleStochastisch**(CPunkt * DerPunkt) (S. 177) , Haltestelle.CStrassennetz**HaltestellenKnoten** (S. 170) @postcondition Die Liste aus **HaltestellenKnoten** (S. 170) ist erzeugt.

Implementiert von **CStrassennetz** (S. 160).

4.59.3.2 void CStrassennetzStochastischUmgebung::setzen-HaltestelleStochastisch (CPunkt * *DerPunkt*)

Ermittelt genau eine Haltestelle nach dem Zufallsprinzip.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CStrassennetzStochastischUmgebung.h**
- **CStrassennetzStochastischUmgebung.cpp**

4.60 CSukzessivVerfahren Klassenreferenz

`#include <CSukzessivVerfahren.h>`

Klassendiagramm für CSukzessivVerfahren:

Öffentliche Datenelemente

- **CSukzessivVerfahren** (int, **CTransportkettengraph** *)
- **~CSukzessivVerfahren** ()

- void **starten** ()
- void **stoppen** ()

4.60.1 Beschreibung der Konstruktoren und Destruktoren

4.60.1.1 CSukzessivVerfahren::CSukzessivVerfahren (int *zusammenhangskomponentenID*, CTransportkettengraph * *transportkettengraph*)

Konstruktor der Klasse Eingabe: ID der Zusammenhangskomponente, fuer das Verfahren angewandt werden soll Weiterhin wird ein Ponter auf dem Transportkettengraph uebergeben

4.60.1.2 CSukzessivVerfahren::~CSukzessivVerfahren ()

Destruktor der Klasse

4.60.2 Dokumentation der Elementfunktionen

4.60.2.1 void CSukzessivVerfahren::starten (void) [virtual]

Beginn der Berechnung der Lokalen Suche

Implementiert von **CLoesungsgenerator** (S. 117).

4.60.2.2 void CSukzessivVerfahren::stoppen () [virtual]

Mit dieser Methode muss es möglich sein, die Berechnung zu beenden.

Implementiert von **CLoesungsgenerator** (S. 117).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSukzessivVerfahren.h**
- **CSukzessivVerfahren.cpp**

4.61 CTestklasse0 Klassenreferenz

`#include <CTestklasse0.h>`

Klassendiagramm für CTestklasse0:

Öffentliche Datenelemente

- **CTestklasse0** (char*)
- **~CTestklasse0** ()
- void **holenDaten** (list<**CPunkt***>*&, list<list<**CPunkt***>*>*&, list<list<**CPunkt***>*>*& wege, list<**CBevoelkerungsdichte-Element***>*&, matrix*&, list<**CQuelleZielListe-Element***>*&, matrix*&, list<**CPendlerListeElement***>*&, list<list<**CPunkt***>*>*&)
- void **holenDaten** (matrix*&, list<**CLinie***>*&, UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>*&)
- matrix* **holenLinienplanQuelleZielMatrix** ()
- list<**CLinie***>* **holenLinienplanVorgegebeneLinien** ()
- UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>* **holen-LinienplanWegenetzgraph** ()
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()
- list<four_tuple<int,int,int,int>*>* **fahrplanFesteVerbindungen** ()
- array<list<three_tuple<int,int,int>*> >* **fahrplan-Zusammenhangskomponenten** ()
- list<**CFahrplanElement***>* **holenFahrplan** ()
- list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- h_array<int,int>* **fahrplanMindesthaltezeit** ()
- h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplan-Mindestumsteigezeit** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- void **holenHaltestellenModulParameter** (int aufruf, int& algorithmus, int& zufallsgeneratorInitialisierung, double& min-Radius, double& maxRadius, double& verbindungsRadiusStadt, double& verbindungsRadiusLand, int& radiustyp, double& haltestellenLage, double& toleranzgrenze, int& bevoelkerungs-Raster, double& koeffizientBevoelkerungsueberversorgung, double& koeffizientBevoelkerungsunterversorgung, double& koeffizient-Flaechenueberversorgung, double& koeffizientFlaechenunterversorgung, double& koeffizientFahrtwunscheinbeziehung, double& koeffizient-Strassenanpassung)
- void **holenLinienplanModulParameter** (int aufruf, int& gesamtlaenge, int& maxLinienLaenge, int& maxLinienAnzahl, double& linienplan-ID, int& geschwindigkeit, int& algorithmus, int& anzeigegraphen, bool& EindeutigeLinienverkn, bool& MehrdeutigeLinienverkn, bool& Aufbruchsverknuepfung, bool& Linienendverknuepfung, bool& Basislinienwahl)
- void **speichernAusschnittStrassennetz** ()

4.61.1 Ausführliche Beschreibung

Testdatenlieferant. Enthaelt einen kleinen uebersichtlichen Satz Testdaten.

Liefert solange Testdaten, bis die entsprechenden Funktionen aus der Datenbank implementiert (d.h. dort ueberschrieben) wurden.

4.61.2 Beschreibung der Konstruktoren und Destruktoren

4.61.2.1 CTestklasse0::CTestklasse0 (char * *db_username*)

4.61.2.2 CTestklasse0::~CTestklasse0 ()

4.61.3 Dokumentation der Elementfunktionen

4.61.3.1 array< three_tuple< int,int,int >* >* CTestklasse0::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Implementiert von **CDatenbank** (S. 31).

4.61.3.2 list< four_tuple< int,int,int,int >* >* CTestklasse0::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Implementiert von **CDatenbank** (S. 32).

4.61.3.3 list< four_tuple< int,int,int,int >*>* CTestklasse0::fahrplanFesteVerbindungen ()

Liefert Feste Verbindungen (Hst-ID, Liniennr.1, Liniennr.2, Hst-ID):

Rückgabe:
Zeiger auf die festen Verbindungen

4.61.3.4 four_tuple< int,int,int,int >* CTestklasse0::fahrplan-Intervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Implementiert von **CDatenbank** (S. 32).

4.61.3.5 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CTestklasse0::fahrplanLinienplan () [virtual]

hole den Linienplan

Rückgabe:
Zeiger auf den Linienplan aus den Testdaten

Implementiert von **CDatenbank** (S. 32).

4.61.3.6 int CTestklasse0::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Implementiert von **CDatenbank** (S. 32).

4.61.3.7 h_array< int,int >* CTestklasse0::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Implementiert von **CDatenbank** (S. 32).

4.61.3.8 h_array< int,list< three_tuple< int,int,int >* >*>* CTestklasse0::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Implementiert von **CDatenbank** (S. 33).

4.61.3.9 array< list< three_tuple< int,int,int >*> >* CTestklasse0::fahrplanZusammenhangskomponenten ()

Liefert Zusammenhangskomponente (Haltest. A, Haltest. B, Liniennummer):

Rückgabe:
Zeiger auf die Liste der Zusammenhangskomponenten

4.61.3.10 void CTestklasse0::holenDaten (matrix *& *verkehrsbedarfsmatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>*& *wegenetzGraph*) [virtual]

hole Daten fuer den Linienplan:

Parameter bedeuten dabei von links nach rechts:

Parameter:

verkehrsbedarfmatrix QuelleZielMatrix

vorgegebeneLinien vorgegebene Linien

wegenetzGraph Wegenetzgraph

Implementiert von **CDatenbank** (S. 34).

4.61.3.11 void CTestklasse0::holenDaten (list< CPunkt *>*& *pflichtHS*, list< list< CPunkt *>*>*& *strassennetz*, list< list< CPunkt *>*>*& *wege*, list< CBevoelkerungsdichteElement *>*& *dichte*, matrix *& *quelleZielMatrix*, list< CQuelleZielListeElement *>*& *quelleZielListe*, matrix *& *pendlerMatrix*, list< CPendlerListeElement *>*& *pendlerListe*, list< list< CPunkt *>*>*& *gebaeude*) [virtual]

Eine ueberladene Funktion. Liefert die angeforderten Werte an die aufrufende Schnittstelle zurueck. Benoetigte Datenstrukturen werden dabei automatisch erzeugt. Daten aus den Datenbanken muessen dabei vor einer Uebergabe an die Module kopiert werden.

Wichtig: Werte, die von der GUI kommen sollen, werden hier NICHT zurueckgegeben

hole Daten fuer die Haltestellen:

Parameter:

pflichtHS Pflichthaltestellen

strassennetz Strassennetz

dichte Bevoelkerungsdichte

quelleZielMatrix Quelle-Ziel-Matrix

quelleZielListe Quelle-Ziel-Liste

pendlerMatrix Pendlermatrix

pendlerListe Pendlerliste

Implementiert von **CDatenbank** (S. 34).

4.61.3.12 list< CFahrplanElement *>* CTestklasse0::holenFahrplan ()

liefert einen Fahrplan:

Rückgabe:

Zeiger auf den Fahrplan

4.61.3.13 void CTestklasse0::holenHaltestellenModulParameter (int *aufruf*, int & *algorithmus*, int & *zufallsgeneratorInitialisierung*, double & *minRadius*, double & *maxRadius*, double & *verbindungsRadiusStadt*, double & *verbindungsRadiusLand*, int & *radiustyp*, double & *haltestellenLage*, double & *toleranzgrenze*, int & *bevoelkerungsRaster*, double & *koeffizientBevoelkerungsueberversorgung*, double & *koeffizientBevoelkerungsunterversorgung*, double & *koeffizientFlaechenueberversorgung*, double & *koeffizientFlaechenunterversorgung*, double & *koeffizientFahrtwunscheinbeziehung*, double & *koeffizientStrassenanpassung*)

Folgende Funktionen dienen der GUI: sie kann auf diese Weise Daten in die Datenbank schieben und auch wieder herausholen

4.61.3.14 void CTestklasse0::holenLinienplanModulParameter (int *aufruf*, int & *gesamtlaenge*, int & *maxLinienLaenge*, int & *maxLinienAnzahl*, double & *linienplanID*, int & *geschwindigkeit*, int & *algorithmus*, int & *anzeigegraphen*, bool & *EindeutigeLinienverkn*, bool & *MehrdeutigeLinienverkn*, bool & *Aufbruchsverknuepfung*, bool & *Linienendverknuepfung*, bool & *Basislinienwahl*) [virtual]

Implementiert von **CDatenbank** (S. 36).

4.61.3.15 matrix * CTestklasse0::holenLinienplanQuelleZielMatrix () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.61.3.16 list< CLinie *>* CTestklasse0::holenLinienplanVorgegebeneLinien ()

4.61.3.17 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CTestklasse0::holenLinienplanWegenetzgraph () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.61.3.18 void CTestklasse0::speichernAusschnittStrassennetz () [virtual]

Implementiert von **CDatenbank** (S. 37).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse0.h**
- **CTestklasse0.cpp**

4.62 CTestklasse1 Klassenreferenz

`#include <CTestklasse1.h>`

Klassendiagramm für CTestklasse1:

Öffentliche Datenelemente

- **CTestklasse1** (char*)
- **~CTestklasse1** ()
- void **holenDaten** (matrix*& , list<**CLinie***>*& , UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>*&)
- matrix* **holenLinienplanQuelleZielMatrix** ()
- list<**CLinie***>* **holenLinienplanVorgegebeneLinien** ()
- UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>* **holenLinienplanWegenetzgraph** ()
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()
- h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplanMindestumsteigezeit** ()
- h_array<int,int>* **fahrplanMindesthaltezeit** ()
- list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- **CGuetemasse*** **holenGuetemasse** (void)

Öffentliche Attribute

- matrix* **qzMatrix**

4.62.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.62.2 Beschreibung der Konstruktoren und Destruktoren

4.62.2.1 CTestklasse1::CTestklasse1 (char * *db_username*)

4.62.2.2 CTestklasse1::~CTestklasse1 ()

4.62.3 Dokumentation der Elementfunktionen

4.62.3.1 array< three_tuple< int,int,int >* >* CTestklasse1::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Implementiert von **CDatenbank** (S. 31).

4.62.3.2 list< four_tuple< int,int,int,int >* >* CTestklasse1::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Implementiert von **CDatenbank** (S. 32).

4.62.3.3 four_tuple< int,int,int,int >* CTestklasse1::fahrplan-Intervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Implementiert von **CDatenbank** (S. 32).

4.62.3.4 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CTestklasse1::fahrplanLinienplan () [virtual]

Implementiert von **CDatenbank** (S. 32).

4.62.3.5 int CTestklasse1::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Implementiert von **CDatenbank** (S. 32).

4.62.3.6 h_array< int,int >* CTestklasse1::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Implementiert von **CDatenbank** (S. 32).

4.62.3.7 h_array< int,list< three_tuple< int,int,int >* >*>* CTestklasse1::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Implementiert von **CDatenbank** (S. 33).

4.62.3.8 void CTestklasse1::holenDaten (matrix *& *quellezielmatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>*& *wegenetzGraph*) [virtual]

hole Daten fuer den Linienplan:

Parameter bedeuten dabei von links nach rechts: QuelleZielMatrix vorgegebene Linien Wegenetzgraph

Implementiert von **CDatenbank** (S. 34).

4.62.3.9 CGuetemasse * CTestklasse1::holenGuetemasse (void) [virtual]

Implementiert von **CDatenbank** (S. 35).

4.62.3.10 matrix * CTestklasse1::holenLinienplanQuelleZielMatrix () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.62.3.11 list< CLinie *>* CTestklasse1::holenLinienplan-VorgegebeneLinien ()

4.62.3.12 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CTestklasse1::holenLinienplanWegenetzgraph () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.62.4 Dokumentation der Datenelemente

4.62.4.1 matrix * CTestklasse1::qzMatrix

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse1.h**
- **CTestklasse1.cpp**

4.63 CTestklasse2 Klassenreferenz

`#include <CTestklasse2.h>`

Klassendiagramm für CTestklasse2:

Öffentliche Datenelemente

- **CTestklasse2** (char*)
- **~CTestklasse2** ()
- void **holenDaten** (matrix*& , list<**CLinie***>*& , UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>*&)
- matrix* **holenLinienplanQuelleZielMatrix** ()
- list<**CLinie***>* **holenLinienplanVorgegebeneLinien** ()
- UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>* **holenLinienplanWegenetzgraph** ()
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()
- h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplanMindestumsteigezeit** ()
- h_array<int,int>* **fahrplanMindesthaltezeit** ()
- list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- **CGuetemasse*** **holenGuetemasse** (void)

Öffentliche Attribute

- matrix* **qzMatrix**

4.63.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.63.2 Beschreibung der Konstruktoren und Destruktoren

4.63.2.1 CTestklasse2::CTestklasse2 (char * *db_username*)

4.63.2.2 CTestklasse2::~CTestklasse2 ()

4.63.3 Dokumentation der Elementfunktionen

4.63.3.1 array< three_tuple< int,int,int >* >* CTestklasse2::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Implementiert von **CDatenbank** (S. 31).

4.63.3.2 list< four_tuple< int,int,int,int >* >* CTestklasse2::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Implementiert von **CDatenbank** (S. 32).

4.63.3.3 four_tuple< int,int,int,int >* CTestklasse2::fahrplan-Intervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Implementiert von **CDatenbank** (S. 32).

4.63.3.4 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CTestklasse2::fahrplanLinienplan () [virtual]

Implementiert von **CDatenbank** (S. 32).

4.63.3.5 int CTestklasse2::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Implementiert von **CDatenbank** (S. 32).

4.63.3.6 h_array< int,int >* CTestklasse2::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Implementiert von **CDatenbank** (S. 32).

4.63.3.7 h_array< int,list< three_tuple< int,int,int >* >*>* CTestklasse2::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Implementiert von **CDatenbank** (S. 33).

4.63.3.8 void CTestklasse2::holenDaten (matrix *& *quellezielmatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>*& *wegenetzGraph*) [virtual]

hole Daten fuer den Linienplan:

Parameter bedeuten dabei von links nach rechts: QuelleZielMatrix vorgegebene Linien Wegenetzgraph

Implementiert von **CDatenbank** (S. 34).

4.63.3.9 CGuetemasse * CTestklasse2::holenGuetemasse (void) [virtual]

Implementiert von **CDatenbank** (S. 35).

4.63.3.10 matrix * CTestklasse2::holenLinienplanQuelleZielMatrix () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.63.3.11 list< CLinie *>* CTestklasse2::holenLinienplanVorgegebeneLinien ()

4.63.3.12 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CTestklasse2::holenLinienplanWegenetzgraph () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.63.4 Dokumentation der Datenelemente

4.63.4.1 matrix * CTestklasse2::qzMatrix

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse2.h**
- **CTestklasse2.cpp**

4.64 CTestklasse3 Klassenreferenz

`#include <CTestklasse3.h>`

Klassendiagramm für CTestklasse3:

Öffentliche Datenelemente

- **CTestklasse3** (char*)
- **~CTestklasse3** ()
- void **holenDaten** (matrix*& , list<**CLinie***>*& , UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>*&)
- matrix* **holenLinienplanQuelleZielMatrix** ()
- list<**CLinie***>* **holenLinienplanVorgegebeneLinien** ()
- UGRAPH<**CWegenetzKnoten***,**CWegenetzKante***>* **holenLinienplanWegenetzgraph** ()
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()

- h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplanMindestumsteigezeit** ()
- h_array<int,int>* **fahrplanMindesthaltezeit** ()
- list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- **CGuetemasse*** **holenGuetemasse** (void)

Öffentliche Attribute

- matrix* **qzMatrix**

4.64.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.64.2 Beschreibung der Konstruktoren und Destruktoren

4.64.2.1 CTestklasse3::CTestklasse3 (char * *db_username*)

4.64.2.2 CTestklasse3::~CTestklasse3 ()

4.64.3 Dokumentation der Elementfunktionen

4.64.3.1 array< three_tuple< int,int,int >* >* CTestklasse3::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Implementiert von **CDatenbank** (S. 31).

4.64.3.2 list< four_tuple< int,int,int,int >* >* CTestklasse3::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Implementiert von **CDatenbank** (S. 32).

4.64.3.3 four_tuple< int,int,int,int >* CTestklasse3::fahrplanIntervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Implementiert von **CDatenbank** (S. 32).

4.64.3.4 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CTestklasse3::fahrplanLinienplan () [virtual]

Implementiert von **CDatenbank** (S. 32).

4.64.3.5 int CTestklasse3::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Implementiert von **CDatenbank** (S. 32).

4.64.3.6 h_array< int,int >* CTestklasse3::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Implementiert von **CDatenbank** (S. 32).

4.64.3.7 h_array< int,list< three_tuple< int,int,int >* >*>* CTestklasse3::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Implementiert von **CDatenbank** (S. 33).

4.64.3.8 void CTestklasse3::holenDaten (matrix *& *quelle-ZielMatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CWegenetzKnoten *,CWegenetzKante *>*& *wegenetzGraph*) [virtual]

hole Daten fuer den Linienplan:

Parameter bedeuten dabei von links nach rechts: QuelleZielMatrix vorgegebene Linien Wegenetzgraph

Implementiert von **CDatenbank** (S. 34).

4.64.3.9 CGuetemasse * CTestklasse3::holenGuetemasse (void) [virtual]

Implementiert von **CDatenbank** (S. 35).

4.64.3.10 matrix * CTestklasse3::holenLinienplanQuelleZielMatrix () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.64.3.11 list< CLinie *>* CTestklasse3::holenLinienplan-VorgegebeneLinien ()

4.64.3.12 UGRAPH< CWegenetzKnoten *,CWegenetzKante *>* CTestklasse3::holenLinienplanWegenetzgraph () [virtual]

Implementiert von **CDatenbank** (S. 36).

4.64.4 Dokumentation der Datenelemente

4.64.4.1 matrix * CTestklasse3::qzMatrix

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse3.h**
- **CTestklasse3.cpp**

4.65 CTestklasse4 Klassenreferenz

`#include <CTestklasse4.h>`

Klassendiagramm für CTestklasse4:

Öffentliche Datenelemente

- **CTestklasse4** (char*)
- **~CTestklasse4** ()
- void **holenDaten** (list<**CPunkt***>*&, list<list<**CPunkt***>*>*&, list<list<**CPunkt***>*>*& wege, list<**CBevoelkerungsdichte-Element***>*&, matrix*&, list<**CQuelleZielListe-Element***>*&, matrix*&, list<**CPendlerListeElement***>*&, list<list<**CPunkt***>*>*&)
- void **speichernAusschnittStrassennetz** ()

4.65.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.65.2 Beschreibung der Konstruktoren und Destruktoren

4.65.2.1 CTestklasse4::CTestklasse4 (char * *db_username*)

4.65.2.2 CTestklasse4::~CTestklasse4 ()

4.65.3 Dokumentation der Elementfunktionen

4.65.3.1 void CTestklasse4::holenDaten (list< CPunkt *>*& *pflicht-haltestellen*, list< list< CPunkt *>*>*& *strassennetz*, list< list< CPunkt *>*>*& *wege*, list< CBevoelkerungsdichteElement *>*& *bevoelkerungsdichte*, matrix *& *quellezielmatrix*, list< CQuelleZiel-ListeElement *>*& *quellezielliste*, matrix *& *pendlermatrix*, list< CPendlerListeElement *>*& *pendlerliste*, list< list< CPunkt *>*>*& *gebaeude*) [virtual]

hole Daten fuer die Haltestellen:

Parameter bedeuten dabei von links nach rechts:

Parameter:

Pflichthaltestellen

Strassennetz

Bevoelkerungsdichte

Quelle-Ziel-Matrix

Quelle-Ziel-Liste

Pendlermatrix

Pendlerliste

Implementiert von **CDatenbank** (S. 34).

4.65.3.2 void CTestklasse4::speichernAusschnittStrassennetz () [virtual]

Implementiert von **CDatenbank** (S. 37).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse4.h**
- **CTestklasse4.cpp**

4.66 CTestklasse5 Klassenreferenz

`#include <CTestklasse5.h>`

Klassendiagramm für CTestklasse5:

Öffentliche Datenelemente

- **CTestklasse5** (char*)
- **~CTestklasse5** ()
- void **holenDaten** (list<**CPunkt***>*&, list<list<**CPunkt***>*>*&, list<list<**CPunkt***>*>*& wege, list<**CBevoelkerungsdichte-Element***>*&, matrix*&, list<**CQuelleZielListe-Element***>*&, matrix*&, list<**CPendlerListeElement***>*&, list<list<**CPunkt***>*>*&)
- void **speichernAusschnittStrassennetz** ()

4.66.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.66.2 Beschreibung der Konstruktoren und Destruktoren

4.66.2.1 CTestklasse5::CTestklasse5 (char * *db_username*)

4.66.2.2 CTestklasse5::~CTestklasse5 ()

4.66.3 Dokumentation der Elementfunktionen

4.66.3.1 void CTestklasse5::holenDaten (list< CPunkt *>*& *pflichthaltestellen*, list< list< CPunkt *>*>*& *strassennetz*, list< list< CPunkt *>*>*& *wege*, list< CBevoelkerungsdichteElement *>*& *bevoelkerungsdichte*, matrix *& *quellezielmatrix*, list< CQuelleZielListeElement *>*& *quellezielliste*, matrix *& *pendlermatrix*, list< CPendlerListeElement *>*& *pendlerliste*, list< list< CPunkt *>*>*& *gebaeude*) [virtual]

hole Daten fuer die Haltestellen:

Parameter bedeuten dabei von links nach rechts:

Parameter:

Pflichthaltestellen

Strassennetz

Bevoelkerungsdichte

Quelle-Ziel-Matrix

Quelle-Ziel-Liste

Pendlermatrix

Pendlerliste

Implementiert von **CDatenbank** (S. 34).

4.66.3.2 void CTestklasse5::speichernAusschnittStrassennetz () [virtual]

Diese Methode schneidet alle Strassen, die in dem durch die Grenzpunkte definierten Rechteck liegen, ab.

Implementiert von **CDatenbank** (S. 37).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse5.h**
- **CTestklasse5.cpp**

4.67 CTestklasse6 Klassenreferenz

`#include <CTestklasse6.h>`

Klassendiagramm für CTestklasse6:

Öffentliche Datenelemente

- **CTestklasse6** (char*)
- **~CTestklasse6** ()
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **fahrplanLinienplan** ()
- h_array<int,list<three_tuple<int,int,int>* >*>* **fahrplan-Mindestumsteigezeit** ()
- h_array<int,int>* **fahrplanMindesthaltezeit** ()
- list<four_tuple<int,int,int,int>* >* **fahrplanDirekterAnschluss** ()
- int **fahrplanMaximaleFahrzeuganzahl** ()
- array<three_tuple<int,int,int>* >* **fahrplanAbfahrtszeiten** ()
- four_tuple<int,int,int,int>* **fahrplanIntervall** ()
- **CGuetemasse*** **holenGuetemasse** (void)

Öffentliche Attribute

- matrix* **qzMatrix**

4.67.1 Ausführliche Beschreibung

Schnittstelle zur Datenbank verarbeitet Datenzugriffe und vermitelt diese auf die Datenquellen (DB, GRASS)

Das Vorgehen ist dabei Folgendes: solange diese einzelnen Funktionen noch nicht implementiert sind, werden automatisch die gleichnamigen Funktionen aus der Basisklasse verwendet, die dann entsprechend ihre Testdaten absondern.

Immer wenn die entsprechenden Punkte implementiert wurden, braucht man sie hier nur zu ueberschreiben

4.67.2 Beschreibung der Konstruktoren und Destruktoren

4.67.2.1 CTestklasse6::CTestklasse6 (char * *db_username*)

4.67.2.2 CTestklasse6::~CTestklasse6 ()

4.67.3 Dokumentation der Elementfunktionen

4.67.3.1 array< three_tuple< int,int,int >* >* CTestklasse6::fahrplanAbfahrtszeiten () [virtual]

holt die Abfahrtszeiten

Rückgabe:
Zeiger auf das LEDA-Array mit den Abfahrtszeiten

Implementiert von **CDatenbank** (S. 31).

4.67.3.2 list< four_tuple< int,int,int,int >* >* CTestklasse6::fahrplanDirekterAnschluss () [virtual]

liefert den direkten Anschluss

Rückgabe:
Zeiger auf die Liste direkter Anschluesse fuer die Testdaten

Implementiert von **CDatenbank** (S. 32).

4.67.3.3 four_tuple< int,int,int,int >* CTestklasse6::fahrplanIntervall () [virtual]

holt das Fahrplanintervall

Rückgabe:
Zeiger auf das Fahrplanintervall

Implementiert von **CDatenbank** (S. 32).

4.67.3.4 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CTestklasse6::fahrplanLinienplan () [virtual]

Implementiert von **CDatenbank** (S. 32).

4.67.3.5 int CTestklasse6::fahrplanMaximaleFahrzeuganzahl () [virtual]

liefert die maximale Fahrzeuganzahl

Rückgabe:
Maximale Fahrzeuganzahl

Implementiert von **CDatenbank** (S. 32).

4.67.3.6 h_array< int,int >* CTestklasse6::fahrplanMindesthaltezeit () [virtual]

holt die Mindesthaltezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit den Mindesthaltezeiten

Implementiert von **CDatenbank** (S. 32).

4.67.3.7 h_array< int,list< three_tuple< int,int,int >* >*>* CTestklasse6::fahrplanMindestumsteigezeit () [virtual]

holt Mindestumsteigezeit

Rückgabe:
Zeiger auf das LEDA-h_array mit der Mindestumsteigezeit

Implementiert von **CDatenbank** (S. 33).

4.67.3.8 CGuetemasse * CTestklasse6::holenGuetemasse (void) [virtual]

Implementiert von **CDatenbank** (S. 35).

4.67.4 Dokumentation der Datenelemente

4.67.4.1 matrix * CTestklasse6::qzMatrix

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTestklasse6.h**
- **CTestklasse6.cpp**

4.68 CTestStrassennetz Klassenreferenz

`#include <Haltestelle.h>`

Klassendiagramm für CTestStrassennetz:

Öffentliche Datenelemente

- void **starten** (void)

4.68.1 Dokumentation der Elementfunktionen

4.68.1.1 void CTestStrassennetz::starten (void) [virtual]

Führt die Berechnungen aus. Kann in den abgeleiteten Klassen erweitert werden. Ansonsten werden die folgenden Schritte ausgeführt:

- Haltestellen setzen,
- Haltestellen an das Strassennetz anpassen,
- Verkehrsbedarfsmatrix erzugen,
- Wegenetzgraph erstellen,
- aufrufen von **CHSSchnittstelle::haltestellenErgebnisrueckgabe** (S. 99) .

Siehe auch:

Programm.CHSSchnittstellehaltestellenErgebnisrueckgabe(GRAPH *,matrix *)

Implementiert von **CStrassennetz** (S. 167).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Haltestelle.h**
- **Haltestelle.cpp**

4.69 CTransportkettengraph Klassenreferenz

`#include <CTransportkettengraph.h>`

Öffentliche Datenelemente

- **CTransportkettengraph** (GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>*)
- **~CTransportkettengraph** ()
- void **setzenStoppenFlag** ()
- void **InitMindestHaltezeit** (int)
- void **InitMindestUmsteigezeit** (int)
- int **gebenFahrzeugAnzahl** ()
- void **berechnenWartezeitResultate** ()
- void **gebenWartezeiten** (double&, double&, double&, double&)
- int **ausgebenMindesthaltezeit** (int)
- void **initialisierenFesterVerbindungen** ()
- void **initialisierenGeruest** ()
- void **setzenMindesthaltezeit** (h_array<int,int> *)
- int **ausgebenMindestumsteigezeit** (int, int, int)
- void **setzenTakt** (int, int)
- void **setzenMindestumsteigezeit** (h_array<int,list< three_-tuple<int,int,int>* >* >*)

- void **setzenNetzbedingteWartezeiten** (h_array<int,int>)
- void **setzenFesterBindungen** (list<four_tuple<int,int,int,int>* >*)
- array<list<three_tuple<int,int,int>* >* >* **berechnenListenDerHaltestellenInAllenZusammenhangskomponenten** ()
- void **berechnenSimplexTableau** (int)
- void **berechnenKnotenPartition** (int)
- array<**CZusammenhangskomponenten**>* **ausgebenZusammenhangskomponenten** ()
- GRAPH<**CKnotenBeschriftung**,**CKantenBeschriftung**>* **liefernZeigerAufGraph** (int)
- void **berechnenMaximalgeruest** (int)
- void **berechnenFahrplan** (array<three_tuple<int,int,int>* >*,four_tuple<int,int,int,int>*)
- list<**CFahrplanElement***>* **gebenFahrplan** ()
- int **gebenZyklomatischeZahl** (int)
- int **gebenAnzahlDerKanten** (int)
- void **startenLokaleSuche** (int)
- void **berechnenBesteLoesung** (int,int)
- h_array<three_tuple<int,int,int>*,int>* **gebenListeNetzbedingtewartezeit** ()
- void **fuellenReisebewertung** (map2<int,int,**CReisebewertung**>&, int)
- double **gebenFahrplanWirkungsgrad** ()
- int **gebenFahrgastfahrten** ()

4.69.1 Ausführliche Beschreibung

In dieser Klasse stehen alle Methoden und Attribute, die für den Transportkettengraphen wichtig sind.

4.69.2 Beschreibung der Konstruktoren und Destruktoren

4.69.2.1 CTransportkettengraph::CTransportkettengraph (GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* *NetzDerTransportverbindungen*)

Nachdem wir von der Funktion starten aus der Klasse **CFahrplan** (S. 55) das Netz der Transportverbindungen erhalten haben, beginnen wir damit, dieses in einen Transportkettengraphen umzubauen. Zu diesem Zweck rufen wir die Funktion umbauenTransporkettengraph auf.

Im Anschluss an den Umbau, rufen wir die Funktion reduzieren-Transportkettengraph auf (erst im zweiten Prototypen). Im Anschluss an diesen Aufruf steht uns ein reduzierter Transportkettengraph zur Verfuegung. Jetzt werden die Zusammenhangskomponenten berechnet. Hierzu wird die Funktion berechnenZusammenhangskomponenten aufgerufen. Dann wird fuer jede dieser Zusammenhangskomponenten die zyklomatische Zahl berechnet. Funktion: berechnenZyklomatischeZahl.

4.69.2.2 CTransportkettengraph::~CTransportkettengraph ()

Destruktor

4.69.3 Dokumentation der Elementfunktionen

4.69.3.1 void CTransportkettengraph::InitMindestHaltezeit (int *mindesthaltezeit*)

Init der Mindesthaltezeit

4.69.3.2 void CTransportkettengraph::InitMindestUmsteigezeit (int *mindestUmsteigezeit*)

Init der Mindestumsteigezeit

4.69.3.3 int CTransportkettengraph::ausgebenMindesthaltezeit (int *haltestellenID*)

Bei Bedarf moechten wir die Mindesthaltezeit einer Haltestelle beruecksichtigen. Wir bekommen von der Programmgruppe ein Hash-Array (LEDA) welches folgende Informationen enthaelt: KEY: Eindeutige ID der Haltestelle (int). VALUE: Mindesthaltezeit in Minuten in Minuten (int), also eventuell gerundet.

4.69.3.4 int CTransportkettengraph::ausgebenMindestumsteigezeit (int *haltestellenID*, int *Liniennummer1*, int *Liniennummer2*)

Diese Funktion gibt die Mindestumsteigezeiten von einer Linie zu einer anderen Linie an einer Haltestelle aus. Zur Speicherung benutzen wir ein Hash-Array (LEDA). Dabei sind die Werte wie folgt in dem Array abgelegt KEY: HaltestellenID. VALUE: Liste von Struct mit folgenden Eintraegen (Liniennummer1, Liniennummer2, min) dabei steigt von Liniennummer1 auf Liniennummer2 um. Die entsprechende Mindestumsteigezeit finden wir wie folgt: Wir hashen auf der ID und erhalten die LIste. Auf diese Liste wenden wir, zumindest im ersten Prototypen eine lineare Suche an, um die entsprechende Zeit zu finden.

4.69.3.5 array< CZusammenhangskomponenten >* CTransportkettengraph::ausgebenZusammenhangskomponenten ()

4.69.3.6 void CTransportkettengraph::berechnenBesteLoesung (int *zusammenhangskomponentenID*, int *baumtiefe*)

Berechnet fuer die Zusammenhangskomponente (1. Parameter) den Entscheidungsbaum bis zur Tiefe (2. Parameter)

4.69.3.7 void CTransportkettengraph::berechnenFahrplan (array< three_tuple< int,int,int >* >* *abfahrtszeiten*, four_tuple< int,int,int,int >* *fahrplanIntervall*)

Diese Methode wird von **CFahrplan** (S. 55) aufgerufen um den Fahrplan zu erstellen. Es werden die in **CFahrplan** (S. 55) geholten Werte fuer die Abfahrtszeiten und FahrplanIntervall uebergeben. Danach wird ueber alle Zusammenhangskomponenten iteriert und dort die Methode berechnenFahrplan (CZusammenhangskomponente) aufgerufen. Dabei wird die Abfahrtszeit der Zusammenhangskomponente und das FahrplanIntervall uebergeben. In CZusammenhangskomponente werden dann die Abfahrtszeiten generiert und eine liste mit CFDahrplanElement Objekten erstellt. Im Anschluss daran, werden die listen der Zusammenhangskomponenten zusammengefuegt.

4.69.3.8 void CTransportkettengraph::berechnenKnotenPartition (int *zusammenhangskomponentenID*)

Bestimmt fuer die Zusammenhangskomponente die Partition Parameter 1: ZusammenhangskomponentenID

Ruft in CZusammenhangskomponente die Methode berechneKnoten-Partiotion()

4.69.3.9 array< list< three_tuple< int,int,int >* >* >* CTransportkettengraph::berechnenListenDerHaltestellenInAllen-Zusammenhangskomponenten ()

Mit dieser Methode berechnet man ein Array, welches aus den Listen der IDs der Haltestellen in jeweils einer Zusammenhangskomponente besteht. Im Prinzip iterieren wir nur ueber alle Zusammenhangskomponenten, dazu haben wir ja das FeldDerZusaammenhangskomponenten und rufen fuer jede der vorhandenen Zusammenhangskomponenten die Methode gebenZusammenhangskomponente-AlsListe in **CZusammenhangskomponenten** (S. 215) auf. Diese Methode liefert uns dann die geforderte Liste, welche wir dann einfach in das neue Array von Listen einhaengen.

4.69.3.10 void CTransportkettengraph::berechnenMaximalgeruest (int *zusammenhangskomponentenID*)

Diese Methode stoesst die Berechnung des Maximalgeruestes innerhalb der angegebenen Zusammenhangskomponente an. Zu diesem Zweck ruft es in der entsprechenden Zusammenhangskomponente, auf die wir mittels des Arrays der Zusammenhangskomponenten zu greifen koennen, die Funktion berechnen-Maximalgeruest auf.

4.69.3.11 void CTransportkettengraph::berechnenSimplexTableau (int *zusammenhangskomponentenID*)

Wir rufen in der entsprechenden Zusammenhangskomponente die Methode berechnenSimplextableau in CZusammenkhangskomponente auf.

4.69.3.12 void CTransportkettengraph::berechnenWartezeit-Resultate ()

Berechnen der Wartezeit-Resultate in allen Zusammenhangskomponenten

4.69.3.13 void CTransportkettengraph::fuellenReisebewertung (map2< int,int,CReisebewertung >& *reisebewertung*, int *anzahl-Haltestellen*)

4.69.3.14 int CTransportkettengraph::gebenAnzahlDerKanten (int)

Diese Methode liefert die Anzahl der Kanten einer Zusammenhangskomponente zurueck

4.69.3.15 int CTransportkettengraph::gebenFahrgastfahrten ()

4.69.3.16 list< CFahrplanElement *>* CTransportkettengraph::gebenFahrplan ()

Diese Methode gibt den erzeugten Fahrplan zurueck

4.69.3.17 double CTransportkettengraph::gebenFahrplan-Wirkungsgrad ()

4.69.3.18 int CTransportkettengraph::gebenFahrzeugAnzahl ()

Geben der Fahrzeuganzahl

4.69.3.19 h_array< three_tuple< int,int,int >*,int >* CTransportkettengraph::gebenListeNetzbedingtewartezeit ()

Gibt von allen Zusammenhangskomponenten ein Hash_array der Netzbedingten Wartezeiten aus.

4.69.3.20 void CTransportkettengraph::gebenWartezeiten (double & *summeNetzbedingteWartezeiten*, double & *wartezeitAufwand*, double & *mindestWartezeitAufwand*, double & *gesamtWartezeit-Aufwand*)

Bestimmung der Summe der Wartezeiten in allen Zusammenhangskomponenten

4.69.3.21 int CTransportkettengraph::gebenZyklomatischeZahl (int *zusammenhangskomponente*)

Diese Methode gibt die Zyklomatische Zahl einer bestimmten Zusammenhangs-komponente zurueck

4.69.3.22 void CTransportkettengraph::initialisierenFesterVerbindungen ()

Diese Funktion iteriert ueber alle Zusammenhangskomponenten und ruft fuer diese die Funktion initialisierenFesterVerbindungen auf

4.69.3.23 void CTransportkettengraph::initialisierenGeruest ()

4.69.3.24 GRAPH< CKnotenBeschriftung,CKantenBeschriftung >* CTransportkettengraph::liefernZeigerAufGraph (int *zusammenhangskomponentenID*)

4.69.3.25 void CTransportkettengraph::setzenFesterBindungen (list< four_tuple< int,int,int,int >* >* *festeBindungen*)

Mit dieser Methode werden die festen Verbindungen zweier Linien gesetzt als four_tupel erhalten wir von **CFahrplan** (S. 55) folgendes: Parameter 1: ZusammenhangskomponentenID Parameter 2: LinienID1 Parameter 3: LinienID2 Parameter 4: HaltestellenID

Wir iterieren ueber die uebergebene Liste und rufen fuer die gegebene Zusammenhangskomponente die Methode setzenFesterBindungen(LinienID1,LinienID2,HaltestellenID) der Klasse CZusammenhangskomponente auf.

4.69.3.26 void CTransportkettengraph::setzenMindesthaltezeit (h_array< int,int >* *mindesthaltezeit*)

Diese Funktion speichert zu Beginn das Hash-Array. Dazu benutzt sie mindestHaltezeit aus dem private Teil. Im Anschluss daran geht die Methode geht wie folgt vor: Sie iteriert ueber alle Zusammenhangskomponenten und uebergibt dort das hash-array an die Methode setzenMindesthaltezeit. Diese schreibt die Mindesthaltezeit dann ebtsprechend dem dort beschriebenen Vorgehen an die Kanten. Fuer naehere Infos siehe setzenMindesthaltezeit in CZusammenhangskomponente.

4.69.3.27 void CTransportkettengraph::setzenMindestumsteigezeit (h_array< int,list< three_tuple< int,int,int >* >* >*)

Diese Funktion speichert das Hash-Array der MindestUmsteigezeiten, welches wir von der Programm- gruppe erhalten haben. Dazu dient mindestUmsteigezeit im private Teil. Im Anschluss daran geht die Methode geht wie folgt vor: Sie iteriert ueber alle Zusammenhangskomponenten und uebergibt dort das hash-array an die Methode setzenMindesthaltezeit. Diese schreibt die Mindesthaltezeit dann ebtsprechend dem dort beschriebenen Vorgehen an die Kanten. Fuer naehere Infos siehe setzenMindesthaltezeit in CZusammenhangskomponente.

4.69.3.28 void CTransportkettengraph::setzenNetzbedingteWartezeiten (h_array< int,int > *netzbedingteWartezeit*)

Diese Funktion

4.69.3.29 void CTransportkettengraph::setzenStoppenFlag ()

setzen des Stoppen Flags

4.69.3.30 void CTransportkettengraph::setzenTakt (int *zusammenhangskomponente*, int *takt*)

Setzen des Taktes in der entsprechenden Zusammenhangskomponente

4.69.3.31 void CTransportkettengraph::startenLokaleSuche (int *zusammenhangskomponentenID*)

Diese Methode startet die Berechnung der Lokalen Suche in der uebergebenen Zusammenhangskomponente

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CTransportkettengraph.h**
- **CTransportkettengraph.cpp**

4.70 CUnsinnigerWertGeliefert Klassenreferenz

`#include <Fehler.h>`

Klassendiagramm für CUnsinnigerWertGeliefert:

Öffentliche Datenelemente

- **CUnsinnigerWertGeliefert** (string)

4.70.1 Ausführliche Beschreibung

Wird geworfen, falls irgendwelche uebergebenen Variablen keine brauchbaren Informationen enthalten

4.70.2 Beschreibung der Konstruktoren und Destruktoren

4.70.2.1 CUnsinnigerWertGeliefert::CUnsinnigerWertGeliefert (string *meldung*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Fehler.h**
- **Fehler.cpp**

4.71 CVerbindung Klassenreferenz

`#include <CVerbindung.h>`

Öffentliche Datenelemente

- **CVerbindung** ()
- **CVerbindung** (node * Start, node * Ziel)
- **~CVerbindung** ()
- void **setzenStart** (node * Start)
- node* **holenStart** (void)
- void **setzenZiel** (node * Start)
- node* **holenZiel** (void)
- void **setzenVerbindung** (list<**CPunkt** *> * Verbindung)
- list<**CPunkt** * >* **holenVerbindung** (void)
- void **setzenLaenge** (double Laenge)
- double **holenLaenge** (void)

4.71.1 Beschreibung der Konstruktoren und Destruktoren

4.71.1.1 CVerbindung::CVerbindung ()

4.71.1.2 CVerbindung::CVerbindung (node * *Start*, node * *Ziel*)

4.71.1.3 CVerbindung::~CVerbindung ()

4.71.2 Dokumentation der Elementfunktionen

4.71.2.1 double CVerbindung::holenLaenge (void)

4.71.2.2 node * CVerbindung::holenStart (void)

4.71.2.3 list< CPunkt * >* CVerbindung::holenVerbindung (void)

4.71.2.4 node * CVerbindung::holenZiel (void)

4.71.2.5 void CVerbindung::setzenLaenge (double *Laenge*)

4.71.2.6 void CVerbindung::setzenStart (node * *Start*)

4.71.2.7 void CVerbindung::setzenVerbindung (list< CPunkt *>* *Verbindung*)

4.71.2.8 void CVerbindung::setzenZiel (node * *Ziel*)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CVerbindung.h**
- **CVerbindung.cpp**

4.72 CVerknuepfungsphase Klassenreferenz

`#include <CVerknuepfungsphase.h>`

Öffentliche Datenelemente

- **CVerknuepfungsphase** (**CNetzplan*** meinNetzplan, matrix* bedarf, **CLPSchnittstelle*** progSchnittstelle)
- **~CVerknuepfungsphase** ()
- void **berechnenVerknuepfung** ()

4.72.1 Ausführliche Beschreibung

Die Verknuepfung zu einem Linienplan entwickelt sich in mehreren Stufen. In jeder Stufe wird solange die jeweils beste Verknuepfung im aktualisieretem **CNetzplan** (S. 121) bestimmt und realisiert, bis keine weitere Verknuepfung in der Stufe mehr moeglich ist.

Realisiert nach Sonntag Seite 97

Weitere Verfahren koennen hier Problemlos integriert werden.

4.72.2 Beschreibung der Konstruktoren und Destruktoren

4.72.2.1 CVerknuepfungsphase::CVerknuepfungsphase (CNetzplan * *meinNetzplan*, matrix * *bedarf*, CLPSchnittstelle * *progSchnittstelle*)

Der Konstruktor der Klasse. Er wird beim generieren der Klasse aufgerufen und sorgt fuer die Initialisierung der Klasse.

Eingabe:

- Ein Zeiger auf ein Objekt der Klasse **CNetzplan** (S. 121). Dieses Objekt enthaelt die fuer die Linienplanung noetigen Daten bereit.

- Eine Zeiger auf die Bedarfsmatrix, die die **CAbschlussverknuepfung** (S. 15) benoetigt.

Rueckgabe:

- nichts

4.72.2.2 CVerknuepfungsphase::~CVerknuepfungsphase ()

Der Destruktor wird aufgerufen, wenn die Klasse geloescht werden soll. In CVerkuepfungsphase fuehrt sie momentan noch nichts aus.

4.72.3 Dokumentation der Elementfunktionen

4.72.3.1 void CVerknuepfungsphase::berechnenVerknuepfung ()

Es wird die Verknuepfungsphase wie sie von Sonntag ab Seite 97 beschrieben wird komplett abgearbeitet. Dazu werden die Methoden in den Klassen **CStartverknuepfung** (S. 153) und CAnschlusverknuepfung aufgerufen. Nach der Abarbeitung befindet sich eine Liste mit Linien im **CNetzplan** (S. 121).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CVerknuepfungsphase.h**
- **CVerknuepfungsphase.cpp**

4.73 CVisualFahrplan Klassenreferenz

`#include <CVisualFahrplan.h>`

Öffentliche Datenelemente

- **CVisualFahrplan** (graph *graphToDraw)
- void **Zeichnen** (int, int)

4.73.1 Ausführliche Beschreibung

Diese Klasse dient der Visualisierung von Graphen

4.73.2 Beschreibung der Konstruktoren und Destruktoren

4.73.2.1 CVisualFahrplan::CVisualFahrplan (graph * *graphToDraw*)

Konstruktor

4.73.3 Dokumentation der Elementfunktionen

4.73.3.1 void CVisualFahrplan::Zeichnen (int *breite*, int *hoehe*)

Zeichnen des Graphen

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CVisualFahrplan.h**
- **CVisualFahrplan.cpp**

4.74 CVisualisierung Klassenreferenz

`#include <CVisualisierung.h>`

Klassendiagramm für CVisualisierung:

Öffentliche Datenelemente

- **CVisualisierung** (const char* titel, int posx = 0, int posy = 0, int breite = 200, int hoehe = 200)
- **~CVisualisierung** ()
- void **zeigenVersorgungsmatrix** (matrix * M)
- void **zeigenMatrixRelativ** (matrix * M, double Maximum)
- void **zeigenFitness** (list<double> * Werte)
- void **zeigenStrassen** (GRAPH<rat_point,rat_segment> * Strassennetz, double Westen, double Norden, double Osten, double Sueden)
- void **zeigenHaltestellen** (list<**CWegenetzKnoten***> * Haltestellen, double Westen, double Norden, double Osten, double Sueden)
- void **zeigenLegende** (void)

4.74.1 Ausführliche Beschreibung

Dient der Visualisierung diverser Kenngoessen innerhalb der Haltestelle. Hierzu zaehlen:

- Bevoelkerungsabdeckung
- Fitnesswerte

4.74.2 Beschreibung der Konstruktoren und Destruktoren

4.74.2.1 CVisualisierung::CVisualisierung (const char * *titel*, int *posx* = 0, int *posy* = 0, int *breite* = 200, int *hoehe* = 200)

Knostruktor, welcher ein Fenster oeffnet.

4.74.2.2 CVisualisierung::~CVisualisierung ()

Destruktor, welcher das Fenster schliesst.

4.74.3 Dokumentation der Elementfunktionen

4.74.3.1 void CVisualisierung::zeigenFitness (list< double >* *Werte*)

Zeigt eine Liste von Fitnesswerten an, wobei selbige durch das hoechste skaliert werden und das Minimum bei Null angenommen wird. Die Punkte werden dabei durch Linien verbunden.

4.74.3.2 void CVisualisierung::zeigenHaltestellen (list< CWegenetz-Knoten *>* *Haltestellen*, double *Westen*, double *Norden*, double *Osten*, double *Sueden*)

4.74.3.3 void CVisualisierung::zeigenLegende (void)

4.74.3.4 void CVisualisierung::zeigenMatrixRelativ (matrix * *M*, double *Maximum*)

4.74.3.5 void CVisualisierung::zeigenStrassen (GRAPH< rat_point,rat_segment >* *Strassennetz*, double *Westen*, double *Norden*, double *Osten*, double *Sueden*)

4.74.3.6 void CVisualisierung::zeigenVersorgungsmatrix (matrix * *M*)

Zeigt die Matrix in Form von Grautoenen. Dabei ist die Kodierung wie folgt: 0 - weiss 1 - sanftes grau 2+ - immer dunkler werdend

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CVisualisierung.h**
- **CVisualisierung.cpp**

4.75 CWegenetzKante Klassenreferenz

`#include <CWegenetzKante.h>`

Öffentliche Datenelemente

- **CWegenetzKante** ()
- **CWegenetzKante** (int Entfernung)
- **CWegenetzKante** (const CWegenetzKante&)
- CWegenetzKante **operator=** (const CWegenetzKante&)
- int **holenEntfernung** ()
- void **setzenEntfernung** (int laenge)
- list<**CPunkt**∗ >∗ **holenKantenListe** ()
- void **setzenKantenListe** (list<**CPunkt**∗>∗ KantenListe)

Freundbeziehungen

- istream& **operator>>** (istream&, CWegenetzKante&)
- ostream& **operator<<** (ostream&, const CWegenetzKante&)

4.75.1 Beschreibung der Konstruktoren und Destruktoren

4.75.1.1 CWegenetzKante::CWegenetzKante ()

4.75.1.2 CWegenetzKante::CWegenetzKante (int *Entfernung*)

Default Konstruktor

4.75.1.3 CWegenetzKante::CWegenetzKante (const CWegenetzKante & *originalBeschriftung*)

Kopier Konstruktor

4.75.2 Dokumentation der Elementfunktionen

4.75.2.1 int CWegenetzKante::holenEntfernung ()

4.75.2.2 list< CPunkt ∗ >∗ CWegenetzKante::holenKantenListe ()

Holen des Strassenzuges, der eine der beiden Kanteninformation darstellt

4.75.2.3 CWegenetzKante CWegenetzKante::operator= (const CWegenetzKante & *originalBeschriftung*)

= Operator

4.75.2.4 void CWegenetzKante::setzenEntfernung (int *laenge*)

4.75.2.5 void CWegenetzKante::setzenKantenListe (list< CPunkt *>* *KantenListe*)

Setzen des Strassenzuges, der eine der beiden Kanteninformation darstellt

4.75.3 Freundbeziehungen und Funktionsdokumentation

4.75.3.1 ostream & operator<< (ostream & *os*, const CWegenetzKante & *beschriftung*) [friend]

<< Operator

4.75.3.2 istream & operator>> (istream & *is*, CWegenetzKante & *beschriftung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CWegenetzKante.h**
- **CWegenetzKante.cpp**

4.76 CWegenetzKnoten Klassenreferenz

`#include <CWegenetzKnoten.h>`

Öffentliche Datenelemente

- **CWegenetzKnoten** (int meineID, int meinePrioritaet)
- **CWegenetzKnoten** (int meineID, int meinePrioritaet, **CPunkt*** meinPunkt)
- **CWegenetzKnoten** (const CWegenetzKnoten&)
- CWegenetzKnoten **operator=** (const CWegenetzKnoten&)
- int **holenKnotenId** ()
- int **holenPrioritaet** ()
- **CPunkt*** **holenPosition** ()
- void **setzenKnotenId** (int meineId)
- void **setzenPrioritaet** (int meinePrioritaet)
- void **setzenPosition** (**CPunkt** * Punkt)
- void **setzenPosition** (double xPosition, double yPosition)

Freundbeziehungen

- istream& **operator>>** (istream&, CWegenetzKnoten&)
- ostream& **operator<<** (ostream&, const CWegenetzKnoten&)

4.76.1 Beschreibung der Konstruktoren und Destruktoren

4.76.1.1 CWegenetzKnoten::CWegenetzKnoten (int *meineID*, int *meinePrioritaet*)

Default Konstruktor

4.76.1.2 CWegenetzKnoten::CWegenetzKnoten (int *meineID*, int *meinePrioritaet*, CPunkt * *meinPunkt*)

Knostruktor mit Punkteuebergabe

4.76.1.3 CWegenetzKnoten::CWegenetzKnoten (const CWegenetzKnoten & *originalBeschriftung*)

Kopier Konstruktor

4.76.2 Dokumentation der Elementfunktionen

4.76.2.1 int CWegenetzKnoten::holenKnotenId ()

4.76.2.2 CPunkt * CWegenetzKnoten::holenPosition ()

4.76.2.3 int CWegenetzKnoten::holenPrioritaet ()

4.76.2.4 CWegenetzKnoten CWegenetzKnoten::operator= (const CWegenetzKnoten & *originalBeschriftung*)

= Operator

4.76.2.5 void CWegenetzKnoten::setzenKnotenId (int *meineId*)

4.76.2.6 void CWegenetzKnoten::setzenPosition (double *xPosition*, double *yPosition*)

4.76.2.7 void CWegenetzKnoten::setzenPosition (CPunkt * *Punkt*)

4.76.2.8 void CWegenetzKnoten::setzenPrioritaet (int *meinePrioritaet*)

4.76.3 Freundbeziehungen und Funktionsdokumentation

4.76.3.1 ostream & operator<< (ostream & *os*, const CWegenetzKnoten & *beschriftung*) [friend]

<< Operator

4.76.3.2 istream & operator>> (istream & *is*, CWegenetzKnoten & *beschriftung*) [friend]

>> Operator

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CWegenetzKnoten.h**
- **CWegenetzKnoten.cpp**

4.77 CZusammenhangskomponenten Klassenreferenz

`#include <CZusammenhangskomponenten.h>`

Öffentliche Datenelemente

- **CZusammenhangskomponenten** (int)
- **~CZusammenhangskomponenten** ()
- void **InitMindestHaltezeit** (int)
- void **setzenStoppenFlag** ()
- void **InitMindestUmsteigezeit** (int)
- list<three_tuple<int,int,int>* >* **gebenZusammenhangskomponenteAlsListe** ()
- void **anfuegenKnoten** (const **CKnotenBeschriftung**&)
- void **anfuegenKante** (int, int, int, int, bool)
- void **ausgebenZGraph** ()
- GRAPH<**CKnotenBeschriftung**,**CKantenBeschriftung**>* **gebenGraphDerZusammenhangskomponte** ()
- void **berechnenSimplexTableau** ()
- void **berechnenZyklomatischeZahl** ()
- void **setzenIdDerZusammenhangskomponente** ()
- int **gebenIdDerZusammenhangskomponente** ()
- int **gebenZyklomatischeZahl** ()
- void **startenLokaleSuche** ()
- void **initialisierenFesterVerbindungen** ()
- void **initialisierenMussInsMaximalgeruest** ()
- void **berechnenMaximalgeruest** ()
- void **setzenMaximalgeruest** (list<edge>*)
- list<edge>* **gebenMaximalgeruest** ()
- void **setzenFesterBindungen** (int,int,int)
- void **berechnenKnotenPartition** ()
- double **gebenNetzbedingteWartezeit** ()
- void **setzenMaximalgeruestKanten** ()
- double **ausgebenMindesthaltezeit** (int)
- void **setzenMindesthaltezeit** (h_array<int, int>*)

- double **ausgebenMindestumsteigezeit** (int,int,int)
- int **gebenFahrzeugAnzahl** ()
- double **gebenSummeNetzbedingteWartezeiten** ()
- double **gebenWartezeitAufwand** ()
- double **gebenFahrplanWirkungsgrad** ()
- double **gebenMindestWartezeitAufwand** ()
- double **gebenGesamtWartezeitAufwand** ()
- void **setzenMindestumsteigezeit** (h_array<int,list< three_tuple<int,int,int>* >* >*)
- void **setzenTakt** (int)
- void **ZeichnenZusammenhangskomponente** ()
- void **berechnenWartezeitResultate** ()
- void **berechnenFahrplan** (three_tuple<int,int,int>*,four_tuple<int,int,int,int>*)
- list<**CFahrplanElement***>* **gebenFahrplan** ()
- void **berechnenBesteLoesung** (int)
- void **initWartezeitResultate** ()
- h_array<three_tuple<int,int,int>*,int>* **gebenListeNetzbedingtewartezeit** ()
- void **fuellenReisebewertung** (map2<int,int,**CReisebewertung**>&)
- int **gebenRelativeGesamtFahrzeit** ()
- int **gebenFahrgastfahrten** ()

4.77.1 Ausführliche Beschreibung

In dieser Klasse beschreibt eine Zusammenhangskomponente und gibt alle Methoden und Attribute an, die zur Berechnung eines Fahrplans dienen.

4.77.2 Beschreibung der Konstruktoren und Destruktoren

4.77.2.1 CZusammenhangskomponenten::CZusammenhangskomponenten (int *zusammenhangskomponentenID*)

Der Konstruktor erhaelt einen int Wert, dieser ist die ID der Zusammenhangskomponente.

4.77.2.2 CZusammenhangskomponenten::~CZusammenhangskomponenten ()

Destruktor

4.77.3 Dokumentation der Elementfunktionen

4.77.3.1 void CZusammenhangskomponenten::InitMindestHaltezeit (int *haltezeit*)

Initialisieren der Mindesthaltezeit auf allen Kanten

4.77.3.2 void CZusammenhangskomponenten::InitMindest-Umsteigezeit (int *haltezeit*)

Initialisieren der Mindestumsteigezeit auf allen Kanten

4.77.3.3 void CZusammenhangskomponenten::Zeichnen-Zusammenhangskomponente ()

Zeichnen der Zusammenhangskomponente

4.77.3.4 void CZusammenhangskomponenten::anfuegenKante (int *knoten1*, int *knoten2*, int *kantenID*, int *verkehrsstrom*, bool *ist-UmsteigeKante*)

4.77.3.5 void CZusammenhangskomponenten::anfuegenKnoten (const CKnotenBeschriftung & *knotenBeschriftung*)

Diese Methode erzeugt einen neuen Knoten und kopiert alle bisherigen Beschriftungen. Weiterhin fuellt diese Methode dann das Hash array zuordnungIdNach-Knoten in CZusammenhangskomponente aus. Dabei notieren wir als Key die interne KnotenID und als Value den entsprechenden Knoten in der Zusammenhangskomponente.

4.77.3.6 double CZusammenhangskomponenten::ausgeben-Mindesthaltezeit (int *haltestellenID*)

4.77.3.7 double CZusammenhangskomponenten::ausgeben-Mindestumsteigezeit (int *haltestellenID*, int *Liniennummer1*, int *Liniennummer2*)

4.77.3.8 void CZusammenhangskomponenten::ausgebenZGraph ()

Gibt den Graph der Zusammenhangskomponente textuell aus

4.77.3.9 void CZusammenhangskomponenten::berechnenBeste-Loesung (int *baumtiefe*)

Berechnet den Entscheidungsbaum bis zur angegebenen Tiefe

4.77.3.10 void CZusammenhangskomponenten::berechnenFahrplan (three_tuple< int,int,int >* *anfangszeit*, four_tuple< int,int,int,int >* *fahrplanIntervall*)

Diese Methode wird aufgerufen, um den Fahrplan fuer die Zusammenhangskomponente erstellen zu koennen. Uebergeben wird die Abfahrtszeit und das Fahrplanintervall. Diese Methode speichert das Fahrplanintervall und ruft die

Private Methode generierenAbfahrtszeit mit der uebergebenen Anfangszeit auf.

4.77.3.11 void CZusammenhangskomponenten::berechnenKnoten-Partition ()

Bei der Wahl der festen Verbindungen kann es passieren, dass der User die Kanten so waehlt, dass die gewaehlten Kanten einen Zyklus ergeben. In einem solchen Fall ist es aber nicht moeglich, die netzbedingten Wartezeiten dieser Verbindungen auf Null zu setzen. Wir brauchen also einen "Zyklentest". Diesen wollen wir mittels einer UNION-FIND-DS realisieren. Folgendes Vorgehen haben wir uns ueberlegt: Beim ersten Aufruf der Funktion wird jeder Knoten in eine eigene Partition gepackt. Dieses erledigt diese Funktion (setzenFester-Bindungen in CZusammenhangskomponenten). Dazu benutzen wir die private Variable knotenPartition. Nach der Initialiserung durch diese Funktion werden wir bei jedem Setzen einer festen Verbindung mittels der Methode setzenFester-Bindungen (dort ist auch der detaillierte Abluaf zu finden) zwei FINDS auf die Knoten der gefundenen Kante durchgefuehrt. Liegen diese in zwei verschiedenen Knotenpartitionen, so erzeugen wir durch die Wahl der Kante als feste Verbindung keinen Zyklus. Tritt dieser Fall ein, so werden alle Knoten der kleineren der beiden Partition in die groessere Knotenpartion "gemergt" (Dieses erledigt LEDA). Befinden sich die Knoten bereits in einer Partition, so wird eine Fehlermeldung ausgegeben.

4.77.3.12 void CZusammenhangskomponenten::berechnen-Maximalgeruest ()

Diese Methode berechnet das von Weigand als Ausgangsloesung geforderte Maximalgeruest. Wir gehen dabei wie folgt vor: Wir uebergeben den Graphen der Zuasmmenhangskomponente (graphDerZusammenhagskomponente), sowie einen Zeiger auf eine Compare-Funktion (vergleichenKantenkosten in CZusammenhangskomponenten) an die Funktion MIN_SPANNING_TREE in LEDA. Als Rueckgabe bekommen wir eine Liste von Kanten, dieses sind die Kanten des Maximalgeruests. Diese Liste uebergeben wir an die Methode setzen-Maximalgeruest in CZusammenhangskomponenten.

4.77.3.13 void CZusammenhangskomponenten::berechnenSimplex-Tableau ()

Wir erzeugen eine Instanz von **CSimplexTableau** (S. 151) und uebergeben an diese die Anzahl Basis und Nichtbasisvariablen, sowie die Taktzeit (Konstruktor). Dann rufen wir die Funktion einfuegenKantenInSimplextableau in **CSimplexTableau** (S. 151) auf, waehrend dieses Aufrufs erhaelt die aufgerufene Methode die Liste aller Kanten der Zusammenhangskomponente.

4.77.3.14 void CZusammenhangskomponenten::berechnen-WartezeitResultate ()

Berechnung der Wartezeit-Resultate in dieser Zusammenhangskomponente

4.77.3.15 void CZusammenhangskomponenten::berechnenZyklomatischeZahl ()

4.77.3.16 void CZusammenhangskomponenten::fuellenReisebewertung (map2< int,int,CReisebewertung >& *reisebewertung*)

4.77.3.17 int CZusammenhangskomponenten::gebenFahrgastfahrten ()

4.77.3.18 list< CFahrplanElement *>* CZusammenhangskomponenten::gebenFahrplan ()

Gibt den Fahrplan als liste von **CFahrplanElement** (S. 58) zurueck

4.77.3.19 double CZusammenhangskomponenten::gebenFahrplanWirkungsgrad ()

Geben-Methode des Fahrplanwirkungsgrades

4.77.3.20 int CZusammenhangskomponenten::gebenFahrzeugAnzahl ()

Geben der FahrzeugAnzahl

4.77.3.21 double CZusammenhangskomponenten::gebenGesamtWartezeitAufwand ()

Geben-Methode

4.77.3.22 GRAPH< CKnotenBeschriftung,CKantenBeschriftung >* CZusammenhangskomponenten::gebenGraphDerZusammenhangskomponte ()

Diese Methode gibt den Graphen der Zusammenhangskomponente an die aufrufende Methode zurueck.

4.77.3.23 int CZusammenhangskomponenten::gebenIdDerZusammenhangskomponente ()

4.77.3.24 h_array< three_tuple< int,int,int >*,int >* CZusammenhangskomponenten::gebenListeNetzbedingtewartezeit ()

Gibt eine Liste zurueck, in der die Netzbedingten Wartezeiten fuer die Kombination LinieVon, Haltestelle, LinieNach stehen.

4.77.3.25 list< edge >* CZusammenhangskomponenten::gebenMaximalgeruest ()

Diese Methode gibt eine Liste von Kanten, welche das Maximalgeruest der entsprechenden Zusammenhangskomponente darstellt, an die aufrufende Methode zurueck.

4.77.3.26 double CZusammenhangskomponenten::gebenMindestWartezeitAufwand ()

Geben-Methode

4.77.3.27 double CZusammenhangskomponenten::gebenNetzbedingteWartezeit ()

Mit dieser Methode wird die netzbedingte Wartezeit der Zusammenhangskomponente ausgegeben. Liegt dieser Wert noch nicht vor, so wird er hier mit der Methode berechnenNetzbedingteWartezeit berechnet.

4.77.3.28 int CZusammenhangskomponenten::gebenRelativeGesamtFahrzeit ()

4.77.3.29 double CZusammenhangskomponenten::gebenSummeNetzbedingteWartezeiten ()

Geben-Methode

4.77.3.30 double CZusammenhangskomponenten::gebenWartezeitAufwand ()

Geben-Methode

4.77.3.31 list< three_tuple< int,int,int >* >* CZusammenhangskomponenten::gebenZusammenhangskomponenteAlsListe ()

Diese Methode gibt als Liste alle Haltestellen der Zusammenhangskomponente zurueck. Um eine Rekonstruktion zu ermoeglichen, sieht ein Listenelement wie folgt aus: Parameter1: HaltestellenID1 Parameter2: HaltestellenID2 Parameter3: Liniennummer Wir erzeugen dabei eine Liste von Tripeln aus dem Standardtyp (Leda) three_tuple. Zu lesen ist das Ganze dann wie folgt: Im Netz der Transportverbindungen ist die Kante gemeint, auf welche die Linie mit der angebenen Nummer von Haltestelle A nach Haltestelle B faehrt.

4.77.3.32 int CZusammenhangskomponenten::gebenZyklomatischeZahl ()

4.77.3.33 void CZusammenhangskomponenten::initWartezeitResultate ()

Initialisiert die Wartezeiten

4.77.3.34 void CZusammenhangskomponenten::initialisierenFesterVerbindungen ()

setzen aller Kantenbeschriftungen des Graphen hinsichtlich festerBindungen auf false

4.77.3.35 void CZusammenhangskomponenten::initialisierenMussInsMaximalgeruest ()

setzen aller Kantenbeschriftungen des Graphen hinsichtlich mussInsMaximalgeruest auf false und die Werte fuer nichtImMaximalgeruest auf false

4.77.3.36 void CZusammenhangskomponenten::setzenFesterBindungen (int *linienID1*, int *linienID2*, int *haltestellenID*)

Mit dieser Methode wird die feste Verbindung zweier Linien gesetzt Parameter 1: LinienID1 Parameter 2: LinienID2 Parameter 3: HaltestellenID

Um die entsprechende Kante, und damit auch Knoten zu finden, benutzen wir die Funktion suchenUmsteigeKante in CZusammenhangskomponenten. Diese gibt die richtige Kante an uns zurueck. Haben wir die Kante erhalten, so geben wir sie an die Funktion setzenUmsteigekante in CZusammenhangskomponente weiter.

4.77.3.37 void CZusammenhangskomponenten::setzenIdDerZusammenhangskomponente ()

4.77.3.38 void CZusammenhangskomponenten::setzenMaximalgeruest (list< edge >* *maximalgeruestKanten*)

Diese Methode speichert die Kanten des Maximalgeruestes in listeMaximalgeruest in CZusammenhangskomponente ab.

4.77.3.39 void CZusammenhangskomponenten::setzenMaximalgeruestKanten ()

Es wird ueber alle Kanten des Maximalgeruestes iteriert. Diese Kanten haben wir ja in der Variable listeMaximalgeruest in CZusammenhangskomponente gespeichert. Fuer alle Kanten in dieser Liste wird in der booleschen Variable imMaximalgeruest, zu finden in **CKantenBeschriftung** (S. 100) mittels der Methode setzenImMaximalgeruest auf TRUE gesetzt.

4.77.3.40 void CZusammenhangskomponenten::setzenMindesthaltezeit (h_array< int,int >* *mindesthaltezeit*)

Diese Methode wird von der Methode setzenMindesthaltezeit in **CTransportkettengraph** (S. 200) aufgerufen. Sie erhaelt das Hash-Array der vom Benutzer eingegebenen Mindesthaltezeiten. Sie macht damit folgendes: Wir iterieren ueber alle Knoten in der Zusammenhangskomponente und lassen uns jeweils HaltestellenID2 des Source Knoten geben. Diese benutzen wir als Key und schauen im erhaltenen Hasharray nach, wie gross die Mindesthaltezeit ist. Im Anschluss daran ermitteln wir alle von diesem Knoten ausgehenden Kanten und uebergeben den Wert an die Methode setzenMindesthaltezeit in **CKantenBeschriftung** (S. 100).

4.77.3.41 void CZusammenhangskomponenten::setzenMindestumsteigezeit (h_array< int,list< three_tuple< int,int,int >* >* >* *mindestumsteigezeit*)

Diese Methode wird von der Methode setzenMindestumsteigezeit in **CTransportkettengraph** (S. 200) aufgerufen. Sie erhaelt das Hash-Array der vom Benutzer eingegebenen Mindestumsteigezeiten. Sie macht damit folgendes: Wir iterieren ueber alle Knoten in der Zusammenhangskomponente. Dann iterieren wir ueber alle out Kanten dieses Knotens, finden wir eine Kante mit istUmsteigekante==true, lesen wir die LinienIDs des Source und Targetknotens mittels ausgebenLiniennummer aus **CKnotenBeschriftung** (S. 104) aus. Dann suchen wir uns im erhaltenen 3-Tupel den Eintrag, welcher den erhaltenen Liniennummern (Reihenfolge Source - Target) entspricht. Der dritte Wert des 3-Tupels ist dann die MindestUmsteigezeit, diese wird an die Kante drangeschrieben. Hierzu dient die Methode setzenMindestumsteigezeit in **CKantenBeschriftung** (S. 100).

4.77.3.42 void CZusammenhangskomponenten::setzenStoppenFlag ()

setzen des Stoppen-Flags

4.77.3.43 void CZusammenhangskomponenten::setzenTakt (int *takt*)

Setzt den Takt

4.77.3.44 void CZusammenhangskomponenten::startenLokaleSuche ()

Diese Methode startet im errechneten Simplextableau die Berechnung der lokalen Suche

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CZusammenhangskomponenten.h**
- **CZusammenhangskomponenten.cpp**

4.78 CKontrolle::fahrplanStruct Strukturenreferenz

`#include <CKontrolle.h>`

Öffentliche Attribute

- **CFPSchnittstelle* lnkCFPSchnittstelle**
- **CFahrplan* lnkFP**
- GRAPH<**CFPlanGraphKnoten***, **CFPlanGraphKante***>* **linienplan**
- int **zusammenhangskomponente**
- int **entscheidungsbaumtiefe**
- **CGUI* lnkCGUI**

4.78.1 Dokumentation der Datenelemente

4.78.1.1 int CKontrolle::fahrplanStruct::entscheidungsbaumtiefe

4.78.1.2 GRAPH< CFPlanGraphKnoten *,CFPlanGraphKante *>* CKontrolle::fahrplanStruct::linienplan

4.78.1.3 CFPSchnittstelle * CKontrolle::fahrplanStruct::lnkCFPSchnittstelle

4.78.1.4 CGUI * CKontrolle::fahrplanStruct::lnkCGUI

4.78.1.5 CFahrplan * CKontrolle::fahrplanStruct::lnkFP

4.78.1.6 int CKontrolle::fahrplanStruct::zusammenhangskomponente

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CKontrolle.h**

4.79 CKontrolle::haltestelleStruct Strukturenreferenz

`#include <CKontrolle.h>`

Öffentliche Attribute

- **CHSSchnittstelle* lnkCHSSchnittstelle**
- **CHaltestelle* lnkHS**
- **CGUI* lnkCGUI**

4.79.1 Ausführliche Beschreibung

Schnittstelle und Modul werden in einer Struktur gespeichert, damit es a) uebersichtlicher wird und b) damit der Datenaustausch ueber die Threads klappt (die Thread-Funktion erlaubt nur ein Argument

4.79.2 Dokumentation der Datenelemente

4.79.2.1 CGUI * CKontrolle::haltestelleStruct::lnkCGUI

4.79.2.2 CHSSchnittstelle * CKontrolle::haltestelleStruct::lnkCHSSchnittstelle

4.79.2.3 CHaltestelle * CKontrolle::haltestelleStruct::lnkHS

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CKontrolle.h**

4.80 leda_cmp_base Klassenreferenz

Klassendiagramm für leda_cmp_base:

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CVergleichsklasse.h**

4.81 CKontrolle::linienplanStruct Strukturenreferenz

`#include <CKontrolle.h>`

Öffentliche Attribute

- **CLPSchnittstelle* lnkCLPSchnittstelle**
- **CLinienplan* lnkLP**
- **CGUI* lnkCGUI**

4.81.1 Dokumentation der Datenelemente

4.81.1.1 CGUI * CKontrolle::linienplanStruct::lnkCGUI

4.81.1.2 CLPSchnittstelle * CKontrolle::linienplanStruct::lnkCLPSchnittstelle

4.81.1.3 CLinienplan * CKontrolle::linienplanStruct::lnkLP

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CKontrolle.h**

4.82 NichtInstanzierteParameter Klassenreferenz

`#include <Fehler.h>`

4.82.1 Ausführliche Beschreibung

Wird bei Auftreten eines nicht instanzierten Parameter(objekt)s verwendet.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Fehler.h**

4.83 sql_cursor Strukturenreferenz

Öffentliche Attribute

- unsigned int **curocn**
- void* **ptr1**
- void* **ptr2**
- unsigned long **magic**

4.83.1 Dokumentation der Datenelemente

4.83.1.1 unsigned int sql_cursor::curocn

4.83.1.2 unsigned long sql_cursor::magic

4.83.1.3 void * sql_cursor::ptr1

4.83.1.4 void * sql_cursor::ptr2

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CDBSchnittstelle.cpp**

4.84 sqlcxp Strukturenreferenz

Öffentliche Attribute

- unsigned short **fillen**
- char **filnam** [20]

4.84.1 Dokumentation der Datenelemente

4.84.1.1 unsigned short sqlcxp::fillen

4.84.1.2 char sqlcxp::filnam[20]

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CDBSchnittstelle.cpp**

4.85 sqlexd Strukturenreferenz

Öffentliche Attribute

- unsigned int **sqlvsn**
- unsigned int **arrsiz**
- unsigned int **iters**
- unsigned int **offset**
- unsigned short **selerr**
- unsigned short **sqlety**
- unsigned int **unused**
- const short* **cud**
- unsigned char* **sqlest**
- const char* **stmt**
- unsigned char** **sqphsv**
- unsigned int* **sqphsl**
- short** **sqpind**
- unsigned int* **sqparm**
- unsigned int** **sqparc**
- unsigned char* **sqhstv** [35]
- unsigned int **sqhstl** [35]
- short* **sqindv** [35]
- unsigned int **sqharm** [35]
- unsigned int* **sqharc** [35]

4.85.1 Dokumentation der Datenelemente

4.85.1.1 unsigned int sqlexd::arrsiz

4.85.1.2 const short * sqlexd::cud

4.85.1.3 unsigned int sqlexd::iters

4.85.1.4 unsigned int sqlexd::offset

4.85.1.5 unsigned short sqlexd::selerr

4.85.1.6 unsigned int * sqlexd::sqharc[35]

4.85.1.7 unsigned int sqlexd::sqharm[35]

4.85.1.8 unsigned int sqlexd::sqhstl[35]

4.85.1.9 unsigned char * sqlexd::sqhstv[35]

4.85.1.10 short * sqlexd::sqindv[35]

4.85.1.11 unsigned char * sqlexd::sqlest

4.85.1.12 unsigned short sqlexd::sqlety

4.85.1.13 unsigned int sqlexd::sqlvsn

4.85.1.14 unsigned int ** sqlexd::sqparc

4.85.1.15 unsigned int * sqlexd::sqparm

4.85.1.16 unsigned int * sqlexd::sqphsl

4.85.1.17 unsigned char ** sqlexd::sqphsv

4.85.1.18 short ** sqlexd::sqpind

4.85.1.19 const char * sqlexd::stmt

4.85.1.20 unsigned int sqlexd::unused

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CDBSchnittstelle.cpp**

4.86 tagTEntry Strukturenreferenz

`#include <CSpVektor.h>`

Öffentliche Attribute

- UINT **Index**
- double **Value**
- tagTEntry* **Next**

4.86.1 Dokumentation der Datenelemente

4.86.1.1 UINT tagTEntry::Index

4.86.1.2 tagTEntry * tagTEntry::Next

4.86.1.3 double tagTEntry::Value

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CSpVektor.h**

4.87 tagTRow Strukturenreferenz

`#include <CSpMatrix.h>`

Öffentliche Attribute

- UINT **Number**
- **TEntry* FirstEntry**
- **TEntry* LastEntry**
- tagTRow* **Next**

4.87.1 Dokumentation der Datenelemente

4.87.1.1 TEntry * tagTRow::FirstEntry

4.87.1.2 TEntry * tagTRow::LastEntry

4.87.1.3 tagTRow * tagTRow::Next

4.87.1.4 UINT tagTRow::Number

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CSpMatrix.h**

4.88 teaCalcFitness Klassenreferenz

Klassendiagramm für teaCalcFitness:

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CStrassennetzES.h**

4.89 TSparseMatrix Klassenreferenz

`#include <CSpMatrix.h>`

Öffentliche Datenelemente

- **TSparseMatrix** ()
- **TSparseMatrix** (UINT m, UINT n)
- **TSparseMatrix** (const TSparseMatrix &M)
- **TSparseMatrix** (UINT m, UINT n, UINT NonZeros, double *Values, UINT *RowPtr, UINT *ColInd)
- **~TSparseMatrix** ()
- TSparseMatrix& **operator=** (const TSparseMatrix &M)
- TSparseMatrix& **operator+=** (const TSparseMatrix &M)
- TSparseMatrix& **Cut** (UINT n)
- TSparseMatrix& **AddIAndMult** (const TSparseMatrix &M, double d)
- TSparseMatrix& **AddVector** (const **TSparseVector** &v, UINT i)
- TSparseMatrix& **AddVectorPos** (const **TSparseVector** &v, UINT i, UINT j)
- void **Compare** (const TSparseMatrix &M, double Epsilon= 0.0) const

- **TRow* AddVector2** (const **TSparseVector** & v, UINT i, **TRow** *StartRow)
- void **InsertEnd** (UINT i, UINT j, double Value)
- **TRow* InsertLineEnd** (UINT i, UINT j, double v, **TRow** *StartRow= NULL)
- void **AddBlock** (TSparseMatrix &M, UINT sx1, UINT sy1, UINT sx2, UINT sy2, UINT dx, UINT dy, double d= 1.0)
- double **Get** (UINT i, UINT j) const
- void **Reset** ()
- void **Reset** (UINT m, UINT n)
- void **ResetRowsWithoutEntries** (UINT m, UINT n)
- TSparseMatrix& **AddScaledMatrix** (double d, const TSparseMatrix &B)
- **TRow* GetFirstRowPtr** () const
- **TRow* GetLastRowPtr** () const
- void **SetFirstRowPtr** (**TRow** *Row)
- void **SetLastRowPtr** (**TRow** *Row)
- **TRow* GetRowPtr** (UINT row)
- double **GetElement** (UINT row, UINT col)
- void **Insert** (UINT row, UINT col, double value)
- void **Update** (UINT row, UINT col, double value)
- void **Delete** (UINT row, UINT col)
- void **DeleteFirstColumn** (void)
- void **DeleteFirstRow** (void)
- **TSparseVector GetFirstRow** (void) const
- UINT **Dim** (UINT i) const
- UINT **NonZeros** () const

Geschützte Attribute

- **TRow** _RowPtr**
- **TRow* _FirstRow**
- **TRow* _LastRow**
- UINT **_m**
- UINT **_n**
- UINT **_NonZeros**

Freundbeziehungen

- class **TSparseVector**
- ostream& **operator<<** (ostream &, const TSparseMatrix &)
- **TSparseVector TransMult** (const **TSparseVector** &, const TSparse-Matrix &)
- TSparseMatrix **operator+** (const TSparseMatrix &, const TSparse-Matrix &)
- void **FastTransMult** (**TSparseVector** &d, const **TSparseVector** &v, const TSparseMatrix &M)

4.89.1 Beschreibung der Konstruktoren und Destruktoren

4.89.1.1 TSparseMatrix::TSparseMatrix ()

4.89.1.2 TSparseMatrix::TSparseMatrix (UINT *m*, UINT *n*)

4.89.1.3 TSparseMatrix::TSparseMatrix (const TSparseMatrix & *M*)

4.89.1.4 TSparseMatrix::TSparseMatrix (UINT *m*, UINT *n*, UINT *NonZeros*, double * *Values*, UINT * *RowPtr*, UINT * *ColInd*)

4.89.1.5 TSparseMatrix::~TSparseMatrix ()

4.89.2 Dokumentation der Elementfunktionen

4.89.2.1 void TSparseMatrix::AddBlock (TSparseMatrix & *M*, UINT *sm1*, UINT *sn1*, UINT *sm2*, UINT *sn2*, UINT *dm*, UINT *dn*, double *d* = 1.0)

4.89.2.2 TSparseMatrix & TSparseMatrix::AddIAndMult (const TSparseMatrix & *M*, double *d*)

4.89.2.3 TSparseMatrix & TSparseMatrix::AddScaledMatrix (double *d*, const TSparseMatrix & *B*)

4.89.2.4 TSparseMatrix & TSparseMatrix::AddVector (const TSparseVector & *v*, UINT *i*)

4.89.2.5 TRow * TSparseMatrix::AddVector2 (const TSparseVector & *v*, UINT *i*, TRow * *StartRow*)

4.89.2.6 TSparseMatrix & TSparseMatrix::AddVectorPos (const TSparseVector & *v*, UINT *i*, UINT *j*)

4.89.2.7 void TSparseMatrix::Compare (const TSparseMatrix & *M*, double *Epsilon* = 0.0) const

4.89.2.8 TSparseMatrix & TSparseMatrix::Cut (UINT *n*)

4.89.2.9 void TSparseMatrix::Delete (UINT *row*, UINT *col*)

4.89.2.10 void TSparseMatrix::DeleteFirstColumn (void)

4.89.2.11 void TSparseMatrix::DeleteFirstRow (void)

4.89.2.12 UINT TSparseMatrix::Dim (UINT *i*) const [inline]

4.89.2.13 double TSparseMatrix::Get (UINT *i*, UINT *j*) const

4.89.2.14 double TSparseMatrix::GetElement (UINT *row*, UINT *col*)

4.89.2.15 TSparseVector TSparseMatrix::GetFirstRow (void) const

4.89.2.16 TRow * TSparseMatrix::GetFirstRowPtr () const [inline]

4.89.2.17 TRow * TSparseMatrix::GetLastRowPtr () const [inline]

4.89.2.18 TRow * TSparseMatrix::GetRowPtr (UINT *row*) [inline]

4.89.2.19 void TSparseMatrix::Insert (UINT *row*, UINT *col*, double *value*)

4.89.2.20 void TSparseMatrix::InsertEnd (UINT *i*, UINT *j*, double *Value*)

4.89.2.21 TRow * TSparseMatrix::InsertLineEnd (UINT *i*, UINT *j*, double *v*, TRow * *StartRow* = NULL)

4.89.2.22 UINT TSparseMatrix::NonZeros () const [inline]

4.89.2.23 void TSparseMatrix::Reset (UINT *m*, UINT *n*)

4.89.2.24 void TSparseMatrix::Reset ()

4.89.2.25 void TSparseMatrix::ResetRowsWithoutEntries (UINT *m*, UINT *n*)

4.89.2.26 void TSparseMatrix::SetFirstRowPtr (TRow * *Row*) [inline]

4.89.2.27 void TSparseMatrix::SetLastRowPtr (TRow * *Row*) [inline]

4.89.2.28 void TSparseMatrix::Update (UINT *row*, UINT *col*, double *value*)

4.89.2.29 TSparseMatrix & TSparseMatrix::operator+= (const TSparseMatrix & *B*)

4.89.2.30 TSparseMatrix & TSparseMatrix::operator= (const TSparseMatrix & *M*)

4.89.3 Freundbeziehungen und Funktionsdokumentation

4.89.3.1 void FastTransMult (TSparseVector & *d*, const TSparseVector & *v*, const TSparseMatrix & *M*) [friend]

4.89.3.2 class TSparseVector [friend]

4.89.3.3 TSparseVector TransMult (const TSparseVector &, const TSparseMatrix &) [friend]

4.89.3.4 TSparseMatrix operator+ (const TSparseMatrix &, const TSparseMatrix &) [friend]

4.89.3.5 ostream& operator<< (ostream & *os*, const TSparseMatrix & *M*) [friend]

4.89.4 Dokumentation der Datenelemente

4.89.4.1 TRow * TSparseMatrix::_FirstRow [protected]

4.89.4.2 TRow * TSparseMatrix::_LastRow [protected]

4.89.4.3 UINT TSparseMatrix::_NonZeros [protected]

4.89.4.4 TRow ** TSparseMatrix::_RowPtr [protected]

4.89.4.5 UINT TSparseMatrix::_m [protected]

4.89.4.6 UINT TSparseMatrix::_n [protected]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSpMatrix.h**
- **CSpMatrix.cpp**

4.90 TSparseVector Klassenreferenz

`#include <CSpVektor.h>`

Öffentliche Datenelemente

- **TSparseVector** ()
- **TSparseVector** (UINT n)
- **TSparseVector** (const TSparseVector &v)
- **~TSparseVector** ()
- TSparseVector& **operator=** (const TSparseVector &v)
- TSparseVector& **operator** *= (const **TSparseMatrix** &M)
- TSparseVector& **operator+=** (const TSparseVector &v)
- TSparseVector& **operator-=** (const TSparseVector &v)
- TSparseVector& **operator** *= (const double d)
- double **operator()** (const UINT d)
- TSparseVector **operator+** (const TSparseVector v)
- TSparseVector **operator-** (const TSparseVector v)
- TSparseVector& **MultAndAdd** (const TSparseVector &v, double d)
- **TEntry*** **GetEntry** () const
- **TEntry*** **GetEndEntry** () const
- void **InsertEnd** (const UINT Index, const double Value)
- void **Add** (const UINT i, const double d)
- void **Set** (const UINT i, const double d)
- void **InsertVector** (UINT StartIndex, const TSparseVector &v)
- void **Reset** (const UINT n= 0)
- UINT **Dim** ()
- UINT **NonZeros** ()

Geschützte Attribute

- **TEntry*** **_Entry**
- **TEntry*** **_EndEntry**
- UINT **_n**
- UINT **_NonZeros**
- **TEntry*** **_CurrentEntry**

Freundbeziehungen

- class **TSparseMatrix**
- ostream& **operator<<** (ostream & os, const TSparseVector & v)
- TSparseVector **TransMult** (const TSparseVector &v, const **TSparseMatrix** &M)
- TSparseVector& **MultAndAdd** (TSparseVector &, const TSparseVector &, double)
- void **FastTransMult** (TSparseVector &d, const TSparseVector &v, const **TSparseMatrix** &M)
- TSparseVector **operator** * (const double d, TSparseVector V)
- TSparseVector **operator** * (TSparseVector V, const double d)

4.90.1 Beschreibung der Konstruktoren und Destruktoren

4.90.1.1 TSparseVector::TSparseVector ()

4.90.1.2 TSparseVector::TSparseVector (UINT *n*)

4.90.1.3 TSparseVector::TSparseVector (const TSparseVector & *v*)

4.90.1.4 TSparseVector::~TSparseVector ()

4.90.2 Dokumentation der Elementfunktionen

4.90.2.1 void TSparseVector::Add (const UINT *i*, const double *d*)

4.90.2.2 UINT TSparseVector::Dim () `[inline]`

4.90.2.3 TEntry * TSparseVector::GetEndEntry () const `[inline]`

4.90.2.4 TEntry * TSparseVector::GetEntry () const `[inline]`

4.90.2.5 void TSparseVector::InsertEnd (const UINT *Index*, const double *Value*)

4.90.2.6 void TSparseVector::InsertVector (UINT *StartIndex*, const TSparseVector & *v*)

4.90.2.7 TSparseVector & TSparseVector::MultAndAdd (const TSparseVector & *v*, double *d*)

4.90.2.8 UINT TSparseVector::NonZeros () [inline]

4.90.2.9 void TSparseVector::Reset (const UINT $n = 0$)

4.90.2.10 void TSparseVector::Set (const UINT i, const double d)

4.90.2.11 TSparseVector & TSparseVector::operator *= (const double d)

4.90.2.12 TSparseVector & TSparseVector::operator *= (const TSparseMatrix & M)

4.90.2.13 double TSparseVector::operator() (const UINT d)

4.90.2.14 TSparseVector TSparseVector::operator+ (const TSparseVector v)

4.90.2.15 TSparseVector & TSparseVector::operator+= (const TSparseVector & v)

4.90.2.16 TSparseVector TSparseVector::operator- (const TSparseVector v)

4.90.2.17 TSparseVector & TSparseVector::operator-= (const TSparseVector & v)

4.90.2.18 TSparseVector & TSparseVector::operator= (const TSparseVector & v)

4.90.3 Freundbeziehungen und Funktionsdokumentation

4.90.3.1 void FastTransMult (TSparseVector & d, const TSparseVector & v, const TSparseMatrix & M) [friend]

4.90.3.2 TSparseVector& MultAndAdd (TSparseVector &, const TSparseVector &, double) [friend]

4.90.3.3 class TSparseMatrix [friend]

4.90.3.4 TSparseVector TransMult (const TSparseVector & v, const TSparseMatrix & M) [friend]

4.90.3.5 TSparseVector operator * (TSparseVector *V*, const double *d*) [friend]

4.90.3.6 TSparseVector operator * (const double *d*, TSparseVector *V*) [friend]

4.90.3.7 ostream& operator<< (ostream & *os*, const TSparseVector & *v*) [friend]

4.90.4 Dokumentation der Datenelemente

4.90.4.1 TEntry * TSparseVector::_CurrentEntry [protected]

4.90.4.2 TEntry * TSparseVector::_EndEntry [protected]

4.90.4.3 TEntry * TSparseVector::_Entry [protected]

4.90.4.4 UINT TSparseVector::_NonZeros [protected]

4.90.4.5 UINT TSparseVector::_n [protected]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **CSpVektor.h**
- **CSpVektor.cpp**

4.91 UnsinnigeErzeugung Klassenreferenz

`#include <Fehler.h>`

4.91.1 Ausführliche Beschreibung

Wird beim nicht korrekten Aufruf eines Konstruktors oder eines nicht erlaubten Konstruktors verwendet.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Fehler.h**

4.92 varchar Strukturenreferenz

Öffentliche Attribute

- unsigned short **len**
- unsigned char **arr** [1]

4.92.1 Dokumentation der Datenelemente

4.92.1.1 unsigned char varchar::arr[1]

4.92.1.2 unsigned short varchar::len

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CDBSchnittstelle.cpp**

4.93 VARCHAR Strukturenreferenz

Öffentliche Attribute

- unsigned short **len**
- unsigned char **arr** [1]

4.93.1 Dokumentation der Datenelemente

4.93.1.1 unsigned char VARCHAR::arr[1]

4.93.1.2 unsigned short VARCHAR::len

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- **CDBSchnittstelle.cpp**

4.94 vergleichenKanten Klassenreferenz

`#include <CVergleichsklasse.h>`

Klassendiagramm für vergleichenKanten:

Öffentliche Datenelemente

- **vergleichenKanten** (const GRAPH<**CKnoten-Beschriftung***,**CKantenBeschriftung***>& g)
- int **operator()** (const edge& kante1, const edge& kante2) const

4.94.1 Beschreibung der Konstruktoren und Destruktoren

4.94.1.1 vergleichenKanten::vergleichenKanten (const GRAPH< CKnotenBeschriftung *,CKantenBeschriftung *>& *g*) [inline]

4.94.2 Dokumentation der Elementfunktionen

4.94.2.1 int vergleichenKanten::operator() (const edge & *kante1*, const edge & *kante2*) const [inline]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CVergleichsklasse.h**

4.95 window Klassenreferenz

Klassendiagramm für window:

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **CVisualisierung.h**

4.96 WohldefinierterAbbruch Klassenreferenz

```
#include <Fehler.h>
```

4.96.1 Ausführliche Beschreibung

Wird zum (halbwegs) kontrollierten Ausstieg aus einem Modul verwendet. Von Gebrauch her aehnlich einem Breakpoint.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Fehler.h**

5 PG369 Datei-Dokumentation

5.1 CAbschlussverknuepfung.cpp Dateireferenz

```
#include "CAbschlussverknuepfung.h"
```

5.2 CAbschlussverknuepfung.h Dateireferenz

```
#include "CNetzplanKnoten.h"
#include "CNetzplanKante.h"
#include "CLinie.h"
#include "CNetzplan.h"
#include "CBasislinie.h"
#include "CAufblinie.h"
#include "Debug.h"
#include <LEDA/matrix.h>
#include <LEDA/array.h>
```

Übersicht

- class **CAbschlussverknuepfung**

5.3 CAufbereitungsphase.cpp Dateireferenz

```
#include "CAufbereitungsphase.h"
```

5.4 CAufbereitungsphase.h Dateireferenz

```
#include "CNetzplanKnoten.h"
#include "CNetzplanKante.h"
#include "CNetzplan.h"
#include "Debug.h"
#include <LEDA/ugraph.h>
#include "LEDA/matrix.h"
#include <LEDA/shortest_path.h>
#include <LEDA/basic_graph_alg.h>
#include <LEDA/edge_map.h>
#include "CWegenetzKnoten.h"
#include "CWegenetzKante.h"
#include <LEDA/graph_misc.h>
#include <math.h>
#include "CFortschritt.h"
```

Übersicht

- class **CAufbereitungsphase**

5.5 CAufblinie.cpp Dateireferenz

```
#include "CAufblinie.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CAufblinie**& ZielLinie)
- ostream& **operator<<** (ostream& os, const **CAufblinie**& Ausgabe-Linie)
- int **compare** (const **CAufblinie**∗ bLinie1, const **CAufblinie**∗ bLinie2)

5.5.1 Dokumentation der Funktionen

5.5.1.1 int compare (const CAufblinie ∗ *bLinie1*, const CAufblinie ∗ *bLinie2*)

5.5.1.2 ostream & operator<< (ostream & *os*, const CAufblinie & *AusgabeLinie*)

<< Operator

5.5.1.3 istream & operator>> (istream & *is*, CAufblinie & *ZielLinie*)

>> Operator

5.6 CAufblinie.h Dateireferenz

```
#include <LEDA/list.h>
#include <iostream.h>
```

Übersicht

- class **CAufblinie**

5.7 CBasislinie.cpp Dateireferenz

```
#include "CBasislinie.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CBasislinie**& zielLinie)
- ostream& **operator<<** (ostream& os, const **CBasislinie**& ausgabeLinie)
- int **compare** (const **CBasislinie**∗ bLinie1, const **CBasislinie**∗ bLinie2)

5.7.1 Dokumentation der Funktionen

5.7.1.1 int compare (const CBasislinie ∗ *bLinie1*, const CBasislinie ∗ *bLinie2*)

Vergleichoperator um von gross nach klein zu sortieren

5.7.1.2 ostream & operator<< (ostream & *os*, const CBasislinie & *ausgabeLinie*)

<< Operator

5.7.1.3 istream & operator>> (istream & *is*, CBasislinie & *zielLinie*)

>> Operator

5.8 CBasislinie.h Dateireferenz

```
#include "CNetzplanKnoten.h"
#include "CNetzplanKante.h"
#include <iostream.h>
```

Übersicht

- class **CBasislinie**

5.9 CBevoelkerungsdichteElement.cpp Dateireferenz

```
#include "CBevoelkerungsdichteElement.h"
```

5.10 CBevoelkerungsdichteElement.h Dateireferenz

```
#include "CPunkt.h"
```

Übersicht

- class **CBevoelkerungsdichteElement**

5.11 CBewertungsphase.cpp Dateireferenz

```
#include "CBewertungsphase.h"
```

5.12 CBewertungsphase.h Dateireferenz

```
#include "../Datenstruktur/CFPlanGraphKnoten.h"
#include "../Datenstruktur/CFPlanGraphKante.h"
#include "../Datenstruktur/CLinie.h"
#include "Debug.h"
#include "CNetzplan.h"
#include "LEDA/graph.h"
#include "LEDA/matrix.h"
#include "LEDA/list.h"
#include "LEDA/edge_map.h"
#include "LEDA/array.h"
#include "LEDA/ugraph.h"
#include "LEDA/shortest_path.h"
#include "LEDA/basic_graph_alg.h"
#include <math.h>
#include "CFortschritt.h"
#include "../Datenstruktur/CGuetemasse.h"
```

Übersicht

- class **CBewertungsphase**

Makrodefinitionen

- #define **BEWERTUNGSPHASE_H**

5.12.1 Makro-Dokumentation

5.12.1.1 #define BEWERTUNGSPHASE_H

5.13 CBezeichner.h Dateireferenz

Übersicht

- class **CBezeichner**

5.14 CBindungen.h Dateireferenz

Übersicht

- class **CBindungen**

5.15 CDatenbank.cpp Dateireferenz

```
#include "CDatenbank.h"
```

5.16 CDatenbank.h Dateireferenz

```
#include "CGRASSSchnittstelle.h"
#include "CDBSchnittstelle.h"
#include "CImport.h"
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
#include "LEDA/array.h"
#include "LEDA/h_array.h"
#include "LEDA/ugraph.h"
#include <LEDA/map2.h>
#include "CPunkt.h"
#include "CBevoelkerungsdichteElement.h"
#include "CFahrplanElement.h"
#include "CQuelleZielListeElement.h"
#include "CPendlerListeElement.h"
#include "CLinie.h"
#include "CFPlanGraphKnoten.h"
#include "CFPlanGraphKante.h"
#include "CWegenetzKnoten.h"
#include "CWegenetzKante.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CDatenbank**

5.17 CDatenstrukturTest.cpp Dateireferenz

```
#include "CFPlanGraphKnoten.h"
```

Funktionen

- int **main** ()

5.17.1 Dokumentation der Funktionen

5.17.1.1 int main ()

5.18 CDatenstrukturTest.h Dateireferenz

5.19 CDBSchnittstelle.cpp Dateireferenz

```
#include "CDBSchnittstelle.h"
#include <sqlca.h>
#include <oraca.h>
#include <sqlda.h>
#include <string.h>
```

Übersicht

- struct **sql_cursor**
- struct **sqlcxp**
- struct **sqlexd**
- struct **varchar**
- struct **VARCHAR**

Typendefinitionen

- typedef struct sql_cursor **sql_cursor**
- typedef struct **sql_cursor SQL_CURSOR**
- typedef void* **sql_context**
- typedef void* **SQL_CONTEXT**

Funktionen

- void **sqlcxt** (void **, unsigned long *, struct **sqlexd** *, const struct **sqlcxp** *)
- void **sqlcx2t** (void **, unsigned long *, struct **sqlexd** *, const struct **sqlcxp** *)

- void **sqlbuft** (void **, char *)
- void **sqlgs2t** (void **, char *)
- void **sqlorat** (void **, unsigned long *, void *)
- void **sqliem** (char *, int *)

5.19.1 Dokumentation der benutzerdefinerten Typen

5.19.1.1 typedef void* SQL_CONTEXT

5.19.1.2 typedef struct sql_cursor SQL_CURSOR

5.19.1.3 typedef void* sql_context

5.19.1.4 typedef struct sql_cursor sql_cursor

5.19.2 Dokumentation der Funktionen

5.19.2.1 void sqlbuft (void **, char *)

5.19.2.2 void sqlcx2t (void **, unsigned long *, struct sqlexd *, const struct sqlcxp *)

5.19.2.3 void sqlcxt (void **, unsigned long *, struct sqlexd *, const struct sqlcxp *)

5.19.2.4 void sqlgs2t (void **, char *)

5.19.2.5 void sqliem (char *, int *)

5.19.2.6 void sqlorat (void **, unsigned long *, void *)

5.20 CDBSchnittstelle.h Dateireferenz

```
#include <stdio.h>
#include <iostream.h>
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
```

```
#include "LEDA/array.h"
#include "LEDA/h_array.h"
#include "LEDA/ugraph.h"
#include "CPunkt.h"
#include "CBevoelkerungsdichteElement.h"
#include "CFahrplanElement.h"
#include "CQuelleZielListeElement.h"
#include "CPendlerListeElement.h"
#include "CLinie.h"
#include "CFPlanGraphKnoten.h"
#include "CFPlanGraphKante.h"
#include "CWegenetzKnoten.h"
#include "CWegenetzKante.h"
#include "CGuetemasse.h"
#include "CFortschritt.h"
#include <new>
```

Übersicht

- class **CDBSchnittstelle**

5.21 CEntscheidungsbaumverfahren.cpp Dateireferenz

```
#include "CEntscheidungsbaumverfahren.h"
```

5.22 CEntscheidungsbaumverfahren.h Dateireferenz

```
#include "CLoesungsgenerator.h"
#include "Debug.h"
#include "CKantenBeschriftung.h"
#include "CKnotenBeschriftung.h"
#include <LEDA/graph.h>
#include <LEDA/tuple.h>
#include <LEDA/h_array.h>
```

Übersicht

- class **CEntscheidungsbaumverfahren**

5.23 CFahrplan.cpp Dateireferenz

```
#include "CFahrplan.h"
#include "CSchnittstellenTest.h"
#include "CFPSchnittstelle.h"
```

Makrodefinitionen

- #define **SCHNITTSTELLE** this → FPSchnittstelle

5.23.1 Makro-Dokumentation

5.23.1.1 #define SCHNITTSTELLE this → FPSchnittstelle

5.24 CFahrplan.h Dateireferenz

```
#include "CHeuristischesLoesungsverfahren.h"
#include "CEntscheidungsbaumverfahren.h"
#include "CSukzessivVerfahren.h"
#include "CLoesungsgenerator.h"
#include "CKnotenBeschriftung.h"
#include "CKantenBeschriftung.h"
#include "CFPlanGraphKante.h"
#include "CFPlanGraphKnoten.h"
#include "CFahrplanElement.h"
#include "CTransportkettengraph.h"
#include <LEDA/tuple.h>
#include <LEDA/graph.h>
#include <LEDA/map2.h>
#include "Debug.h"
#include "Fehler.h"
#include "CReisebewertung.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CFahrplan**

5.25 CFahrplanElement.cpp Dateireferenz

```
#include "CFahrplanElement.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CFahrplanElement**& original-Fahrplan)
- ostream& **operator<<** (ostream& os, const **CFahrplanElement**& originalFahrplan)

5.25.1 Dokumentation der Funktionen

5.25.1.1 ostream & operator<< (ostream & *os*, const CFahrplan-Element & *originalFahrplan*)

<< Operator

5.25.1.2 istream & operator>> (istream & *is*, CFahrplanElement & *originalFahrplan*)

>> Operator

5.26 CFahrplanElement.h Dateireferenz

```
#include <iostream.h>
```

Übersicht

- class **CFahrplanElement**

5.27 CFahrplanTest.cpp Dateireferenz

```
#include "CTransportkettengraph.h"
#include <LEDA/graph.h>
#include "CFPlanGraphKante.h"
#include "CSchnittstellenTest.h"
#include "CFPlanGraphKnoten.h"
#include <string.h>
```

Funktionen

- int **main** (int argc, char* argv[])

5.27.1 Dokumentation der Funktionen

5.27.1.1 int main (int *argc*, char * *argv*[])

5.28 CFahrplanTest.h Dateireferenz

5.29 CFortschritt.cpp Dateireferenz

```
#include <sys/time.h>
#include "CFortschritt.h"
#include "Debug.h"
```

Makrodefinitionen

- #define **HINTERGRUND** 220,220,220
- #define **VORDERGRUND** 0,0,0
- #define **INTERVALL_ZEIT** 3000
- #define **INTERVALL_FORTSCHRITT** 100

Funktionen

- color **bunt** (int prozent)
- unsigned long **getcurrenttime** (timeval *zeit)

5.29.1 Makro-Dokumentation

5.29.1.1 #define HINTERGRUND 220,220,220

5.29.1.2 #define INTERVALL_FORTSCHRITT 100

5.29.1.3 #define INTERVALL_ZEIT 3000

5.29.1.4 #define VORDERGRUND 0,0,0

5.29.2 Dokumentation der Funktionen

5.29.2.1 color bunt (int *prozent*)

5.29.2.2 unsigned long getcurrenttime (timeval * *zeit*)

5.30 CFortschritt.h Dateireferenz

```
#include <LEDA/rat_point.h>
#include <LEDA/window.h>
#include <LEDA/list.h>
#include "Debug.h"
```

Übersicht

- class **CFortschritt**

Makrodefinitionen

- #define **NR_SAMPLES** 30
- #define **LAENGE** 100

5.30.1 Makro-Dokumentation

5.30.1.1 #define LAENGE 100

5.30.1.2 #define NR_SAMPLES 30

5.31 CFPlanGraphKante.cpp Dateireferenz

```
#include "CFPlanGraphKante.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CFPlanGraphKante**& beschriftung)
- ostream& **operator<<** (ostream& os, const **CFPlanGraphKante**& beschriftung)

5.31.1 Dokumentation der Funktionen

5.31.1.1 ostream & operator<< (ostream & *os*, const CFPlanGraphKante & *beschriftung*)

<< Operator

5.31.1.2 istream & operator>> (istream & *is*, CFPlanGraphKante & *beschriftung*)

>> Operator

5.32 CFPlanGraphKante.h Dateireferenz

```
#include <iostream.h>
```

Übersicht

- class **CFPlanGraphKante**

5.33 CFPlanGraphKnoten.cpp Dateireferenz

```
#include "CFPlanGraphKnoten.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CFPlanGraphKnoten**& beschriftung)
- ostream& **operator<<** (ostream& os, const **CFPlanGraphKnoten**& beschriftung)

5.33.1 Dokumentation der Funktionen

5.33.1.1 ostream & operator<< (ostream & *os*, const CFPlanGraphKnoten & *beschriftung*)

<< Operator

5.33.1.2 istream & operator>> (istream & *is*, CFPlanGraphKnoten & *beschriftung*)

>> Operator

5.34 CFPlanGraphKnoten.h Dateireferenz

```
#include <iostream.h>
```

Übersicht

- class **CFPlanGraphKnoten**

5.35 CFPSchnittstelle.cpp Dateireferenz

```
#include "CFPSchnittstelle.h"
```

5.36 CFPSchnittstelle.h Dateireferenz

```
#include "CSchnittstelleMitGUI.h"
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
#include "LEDA/array.h"
#include "LEDA/h_array.h"
#include "LEDA/ugraph.h"
#include "../Datenstruktur/CFahrplanElement.h"
#include "../Datenstruktur/CGuetemasse.h"
#include "CGUI.h"
```

Übersicht

- class **CFPSchnittstelle**

5.37 CGRASSSchnittstelle.cpp Dateireferenz

```
#include "CGRASSSchnittstelle.h"
#include "stdio.h"
```

5.38 CGRASSSchnittstelle.h Dateireferenz

```
#include "../Datenstruktur/define.h"
#include "CFortschritt.h"
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
#include "LEDA/array.h"
#include "LEDA/h_array.h"
#include "LEDA/ugraph.h"
```

```
#include "CPunkt.h"
#include "CKnoten.h"
#include "../Datenstruktur/Debug.h"
#include <stdio.h>
#include <stdlib.h>
#include "gis.h"
#include "Vect.h"
#include "site.h"
#include "../Datenstruktur/CFPlanGraphKnoten.h"
#include "../Datenstruktur/CFPlanGraphKante.h"
#include "../Datenstruktur/CWegenetzKnoten.h"
#include "../Datenstruktur/CWegenetzKante.h"
#include "../Datenstruktur/CLinie.h"
#include "../Datenstruktur/CLeda.h"
```

Übersicht

- class **CGRASSSchnittstelle**

5.39 CGuetemasse.cpp Dateireferenz

```
#include "CGuetemasse.h"
```

5.40 CGuetemasse.h Dateireferenz

```
#include <LEDA/string.h>
#include <LEDA/h_array.h>
#include <LEDA/tuple.h>
#include <LEDA/map2.h>
#include <CReisebewertung.h>
#include <LEDA/matrix.h>
#include "Debug.h"
```

Übersicht

- class **CGuetemasse**

5.41 CGUI.cpp Dateireferenz

```
#include "CGUI.h"
#include <cstring>
```

Makrodefinitionen

- #define **EXTERN** extern
- #define **ET_OK** 0
- #define **ET_ERROR** 1
- #define **UNIX** 1
- #define **WIN32** 0
- #define **EXTERN** extern

Funktionen

- EXTERN int **Et_Eval** (const char *,int,const char*,...)
- EXTERN int **Et_EvalInt** (const char *,int,const char*,...)
- EXTERN char* **Et_EvalString** (const char *,int,const char*,...)
- EXTERN double **Et_EvalDouble** (const char *,int,const char*,...)
- EXTERN void **Et_EvalInclude** (const char *,char *)
- EXTERN void **Et_Init** (int *, char **)
- EXTERN void **Et_MainLoop** (void)
- EXTERN void **Et_ReadStdin** (void)
- array<three_tuple<int,int,int>*> **int** * (0)
- four_tuple<int,int,int,int> **int** (0,0,23,59)
- int **Length** (char *str)
- EXTERN void **Et_InstallCommand** (char *,int (*)(void*,struct Tcl_Interp*,int,char**))
- void **Et_Init_cgui_et** (void)

Variablen

- EXTERN struct Tcl_Interp* **Et_Interp**
- EXTERN struct Tk_Window_* **Et_MainWindow**
- EXTERN struct _XDisplay* **Et_Display**
- **CKontrolle* globalkontrolle**
- **CGUISchnittstelle* globalschnittstelle**
- array<three_tuple<int,int,int>*>* **Abfahrtszeiten** = new array<three_tuple<int
- array<three_tuple<int,int,int>*> **int**
- four_tuple<int,int,int,int>* **FahrplanIntervall** = new four_tuple<int
- h_array<int,int>* **Mindesthaltezeiten** = new h_array<int
- h_array<int,list<three_tuple<int,int,int>* >*>* **Mindestumsteigezeiten** = new h_array<int

- h_array<int,list<three_tuple<int,int,int>* >*> **list<three_tuple<int,int,int>*>***
- list<four_tuple<int,int,int,int>* >* **DirekteAnschluesse** = new list<four_tuple<int
- list<four_tuple<int,int,int,int>* > **int***

5.41.1 Makro-Dokumentation

5.41.1.1 #define ET_ERROR 1

5.41.1.2 #define ET_OK 0

5.41.1.3 #define EXTERN extern

5.41.1.4 #define EXTERN extern

5.41.1.5 #define UNIX 1

5.41.1.6 #define WIN32 0

5.41.2 Dokumentation der Funktionen

5.41.2.1 EXTERN int Et_Eval (const char *, int, const char *, ...)

5.41.2.2 EXTERN double Et_EvalDouble (const char *, int, const char *, ...)

5.41.2.3 EXTERN void Et_EvalInclude (const char *, char *)

5.41.2.4 EXTERN int Et_EvalInt (const char *, int, const char *, ...)

5.41.2.5 EXTERN char* Et_EvalString (const char *, int, const char *, ...)

5.41.2.6 EXTERN void Et_Init (int *, char **)

5.41.2.7 void Et_Init_cgui_et (void)

5.41.2.8 EXTERN void Et_InstallCommand (char *, int(*)(void *,struct Tcl_Interp *,int,char **))

5.41.2.9 EXTERN void Et_MainLoop (void)

5.41.2.10 EXTERN void Et_ReadStdin (void)

5.41.2.11 int Length (char * *str*)

5.41.2.12 four_tuple<int,int,int,int> int (0, 0, 23, 59)

5.41.2.13 array<three_tuple<int,int,int>*> int * (0)

5.41.3 Variablen-Dokumentation

5.41.3.1 array< three_tuple< int,int,int >* >* Abfahrtszeiten = new array<three_tuple<int

5.41.3.2 list< four_tuple< int,int,int,int >* >* DirekteAnschluesse = new list<four_tuple<int

5.41.3.3 EXTERN struct _XDisplay * Et_Display

5.41.3.4 EXTERN struct Tcl_Interp * Et_Interp

5.41.3.5 EXTERN struct Tk_Window_ * Et_MainWindow

5.41.3.6 four_tuple< int,int,int,int >* FahrplanIntervall = new four_tuple<int

5.41.3.7 h_array< int,int >* Mindesthaltezeiten = new h_array<int

5.41.3.8 h_array< int,list< three_tuple< int,int,int >* >*>* Mindestumsteigezeiten = new h_array<int

5.41.3.9 CKontrolle * globalkontrolle

5.41.3.10 CGUISchnittstelle * globalschnittstelle

5.41.3.11 list< four_tuple< int,int,int,int >* > int

5.41.3.12 list<four_tuple<int,int,int,int>* > int*

5.41.3.13 h_array<int,list<three_tuple<int,int,int>* >*> list<three_tuple<int,int,int>*>*

5.42 CGUI.h Dateireferenz

```
#include "CKontrolle.h"
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/string.h"
#include "CQuelleZielListeElement.h"
```

Übersicht

- class **CGUI**

5.43 CGUISchnittstelle.cpp Dateireferenz

```
#include "CGUISchnittstelle.h"
```

5.44 CGUISchnittstelle.h Dateireferenz

```
#include "LEDA/graph.h"
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
#include "LEDA/array.h"
#include "LEDA/h_array.h"
#include "LEDA/ugraph.h"
#include "CSchnittstelle.h"
#include "CDatenbank.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CGUISchnittstelle**

5.45 CHaltestelle.cpp Dateireferenz

```
#include "CHaltestelle.h"
#include "CStrassennetzStochastisch.h"
#include "CStrassennetzStochastischUmgebung.h"
#include "CStrassennetzES.h"
#include "../Datenstruktur/CFortschritt.h"
```

5.46 CHaltestelle.h Dateireferenz

```
#include "CStrassennetz.h"
#include "../Programm/CHSSchnittstelle.h"
```

Übersicht

- class **CHaltestelle**

5.47 CHeuristischesLoesungsverfahren.cpp Dateireferenz

```
#include "CHeuristischesLoesungsverfahren.h"
```

5.48 CHeuristischesLoesungsverfahren.h Dateireferenz

```
#include <LEDA/graph.h>
#include "CLoesungsgenerator.h"
```

Übersicht

- class **CHeuristischesLoesungsverfahren**

5.49 CHSSchnittstelle.cpp Dateireferenz

```
#include <fstream.h>
#include "CHSSchnittstelle.h"
#include "CGUI.h"
```

5.50 CHSSchnittstelle.h Dateireferenz

```
#include "CSchnittstelleMitGUI.h"
#include "LEDA/matrix.h"
```

```
#include "LEDA/list.h"
#include "LEDA/graph.h"
#include "../Datenstruktur/CPunkt.h"
#include "../Datenstruktur/CBevoelkerungsdichteElement.h"
#include "../Datenstruktur/CQuelleZielListeElement.h"
#include "../Datenstruktur/CPendlerListeElement.h"
#include "../Datenstruktur/CWegenetzKnoten.h"
#include "../Datenstruktur/CParameterHaltestelle.h"
#include "../Datenstruktur/CGuetemasse.h"
#include "../Datenstruktur/CFortschritt.h"
#include "CDatenbank.h"
```

Übersicht

- class **CHSSchnittstelle**

5.51 CImport.cpp Dateireferenz

```
#include "CImport.h"
```

5.52 CImport.h Dateireferenz

Übersicht

- class **CImport**

5.53 CKantenBeschriftung.cpp Dateireferenz

```
#include "CKantenBeschriftung.h"
```

Funktionen

- istream& **operator>>** (istream& instream, **CKantenBeschriftung**& beschriftung)
- ostream& **operator<<** (ostream& outstream, const **CKantenBeschriftung**& beschriftung)

5.53.1 Dokumentation der Funktionen

5.53.1.1 ostream & operator<< (ostream & *outstream*, const CKantenBeschriftung & *beschriftung*)

<< Operator

5.53.1.2 istream & operator>> (istream & *instream*, CKantenBeschriftung & *beschriftung*)

>> Operator

5.54 CKantenBeschriftung.h Dateireferenz

```
#include <iostream.h>
```

Übersicht

- class **CKantenBeschriftung**

5.55 CKnoten.h Dateireferenz

Übersicht

- class **CKnoten**

5.56 CKnotenBeschriftung.cpp Dateireferenz

```
#include "CKnotenBeschriftung.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CKnotenBeschriftung**& beschriftung)
- ostream& **operator<<** (ostream& os, const **CKnotenBeschriftung**& beschriftung)

5.56.1 Dokumentation der Funktionen

5.56.1.1 ostream & operator<< (ostream & *os*, const CKnotenBeschriftung & *beschriftung*)

5.56.1.2 istream & operator>> (istream & *is*, CKnotenBeschriftung & *beschriftung*)

5.57 CKnotenBeschriftung.h Dateireferenz

```
#include <iostream.h>
```

Übersicht

- class **CKnotenBeschriftung**

Makrodefinitionen

- #define **CKNOTENBESCHRIFTUNG_H**

5.57.1 Makro-Dokumentation

5.57.1.1 #define CKNOTENBESCHRIFTUNG_H

5.58 CKontrolle.cpp Dateireferenz

```
#include "../Datenstruktur/Debug.h"
#include "CKontrolle.h"
#include <unistd.h>
```

5.59 CKontrolle.h Dateireferenz

```
#include <pthread.h>
#include <signal.h>
#include <iostream>
#include <new>
#include "CHSSchnittstelle.h"
#include "CLPSchnittstelle.h"
#include "CFPSchnittstelle.h"
#include "CGUISchnittstelle.h"
#include "CDatenbank.h"
#include "CTestklasse0.h"
#include "CTestklasse1.h"
#include "CTestklasse2.h"
#include "CTestklasse3.h"
#include "CTestklasse4.h"
#include "CTestklasse5.h"
#include "../Fahrplan/CFahrplan.h"
#include "../Haltestelle/CHaltestelle.h"
#include "../Linienplan/CLinienplan.h"
```

```
#include "../Datenstruktur/Fehler.h"
```

Übersicht

- class **CKontrolle**
- struct **CKontrolle::fahrplanStruct**
- struct **CKontrolle::haltestelleStruct**
- struct **CKontrolle::linienplanStruct**

5.60 CLeda.cpp Dateireferenz

```
#include "CLeda.h"
```

Funktionen

- bool **pruefenKnotenverbindung** (GRAPH<void *, void * > * DerGraph, node * knotenVon, node * knotenNach)
- edge* **holenKnotenverbindung** (GRAPH<void *, void * > * DerGraph, node * knotenVon, node * knotenNach)
- void **holenKnoten** (GRAPH<void *, void * > * DerGraph, edge * AktuelleKante, node * Start, node * Ziel)

5.60.1 Dokumentation der Funktionen

5.60.1.1 void holenKnoten (GRAPH< void *,void * >* *DerGraph*, edge * *AktuelleKante*, node * *Start*, node * *Ziel*)

5.60.1.2 edge * holenKnotenverbindung (GRAPH< void *,void * >* *DerGraph*, node * *knotenVon*, node * *knotenNach*)

Gibt die Verbindung zwischen zwei Knoten zurueck, falls vorhanden.

Parameter:

DerGraph Der Graph, aus welchem die Knoten entstammen.

knotenVon Knoten, von welchem aus auf eine Kante geprueft werden soll.

knotenNach Knoten, zu welchem auf eine Kante geprueft werden soll.

Rückgabe:

Eine edge, falls die Kante existiert, NULL sonst.

5.60.1.3 bool pruefenKnotenverbindung (GRAPH< void *,void * >* *DerGraph*, node * *knotenVon*, node * *knotenNach*)

Gibt zurueck, ob zwischen zwei gegeben Knoten ausgehend von knotenVon zu knotenNach eine Kante existiert.

Parameter:

DerGraph Der Graph, aus welchem die Knoten entstammen.

knotenVon Knoten, von welchem aus auf eine Kante geprueft werden soll.

knotenNach Knoten, zu welchem auf eine Kante geprueft werden soll.

Rückgabe:

true, falls eine Kante existiert, false sonst.

5.61 CLeda.h Dateireferenz

```
#include <LEDA/graph.h>
```

Funktionen

- void **holenKnoten** (edge* AktuelleKante, node * Start, node * Ziel)

5.61.1 Dokumentation der Funktionen

5.61.1.1 void holenKnoten (edge * *AktuelleKante*, node * *Start*, node * *Ziel*)

Gibt die zugehoerigen Knoten zur uebergebenen Kante zurueck.

5.62 CLinie.cpp Dateireferenz

```
#include "CLinie.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CLinie**& ZielLinie)
- ostream& **operator<<** (ostream& os, const **CLinie**& AusgabeLinie)

5.62.1 Dokumentation der Funktionen

5.62.1.1 ostream & operator<< (ostream & *os*, const CLinie & *AusgabeLinie*)

<< Operator

5.62.1.2 istream & operator>> (istream & *is*, CLinie & *ZielLinie*)

>> Operator

5.63 CLinie.h Dateireferenz

```
#include <LEDA/list.h>
#include <iostream.h>
#include "../Linienplan/CNetzplan.h"
```

Übersicht

- class **CLinie**

5.64 CLinienplan.cpp Dateireferenz

```
#include "CLinienplan.h"
#include "../Datenstruktur/define.h"
```

5.65 CLinienplan.h Dateireferenz

```
#include "CBewertungsphase.h"
#include "CVerknuepfungsphase.h"
#include "CAufbereitungsphase.h"
#include "CNetzplan.h"
#include "CWegenetzKnoten.h"
#include "CWegenetzKante.h"
#include "Debug.h"
#include "../Programm/CLPSchnittstelle.h"
#include "CLPSchnittstelleTest.h"
#include "../Datenstruktur/CFPlanGraphKante.h"
#include "../Datenstruktur/CFPlanGraphKnoten.h"
#include <LEDA/matrix.h>
#include <LEDA/graphwin.h>
#include <LEDA/color.h>
#include <LEDA/window.h>
```

Übersicht

- class **CLinienplan**

5.66 CLinienplanTest.cpp Dateireferenz

```
#include "CLinienplanTest.h"
```

Funktionen

- void **jinhatest_bewertung** ()
- void **jinhatest** ()
- void **michaeltestsv** ()
- void **michaeltestvk** ()
- void **michaeltest** ()
- void **gesamtmodultestfahrplan** ()
- void **gesamtmodultesttim** (int argc, char* argv[])
- void **help** (void)
- void **hauptmenu** ()

5.66.1 Dokumentation der Funktionen

5.66.1.1 void gesamtmodultestfahrplan ()

5.66.1.2 void gesamtmodultesttim (int *argc*, char * *argv*[])

5.66.1.3 void hauptmenu ()

5.66.1.4 void help (void)

5.66.1.5 void jinhatest ()

5.66.1.6 void jinhatest_bewertung ()

5.66.1.7 void michaeltest ()

5.66.1.8 void michaeltestsv ()

5.66.1.9 void michaeltestvk ()

5.67 CLinienplanTest.h Dateireferenz

```
#include "CLinienplan.h"
#include "CLinienplanTestReza.h"
#include "../Programm/CLPSchnittstelle.h"
#include "../Programm/CGUISchnittstelle.h"
#include "../Programm/CKontrolle.h"
#include "Debug.h"
#include <iostream.h>
```

5.68 CLinienplanTestReza.cpp Dateireferenz

```
#include "CLinienplanTestReza.h"
```

Funktionen

- void **holenDaten** (matrix*& verkehrsbedarfsmatrix, list<**CLinie***>*& vorgegebeneLinien, UGRAPH<**CNetzplanKnoten***,**CNetzplanKante***>*& netzplangraph)
- void **rezatest** ()

5.68.1 Dokumentation der Funktionen

5.68.1.1 void holenDaten (matrix *& *verkehrsbedarfsmatrix*, list< CLinie *>*& *vorgegebeneLinien*, UGRAPH< CNetzplanKnoten *,CNetzplanKante *>*& *netzplangraph*)

5.68.1.2 void rezatest ()

5.69 CLinienplanTestReza.h Dateireferenz

```
#include "CAbschlussverknuepfung.h"
#include "Debug.h"
#include <iostream.h>
#include "LEDA/list.h"
#include "LEDA/matrix.h"
#include "LEDA/integer_matrix.h"
#include "LEDA/tuple.h"
#include "LEDA/array.h"
#include "LEDA/h_array.h"
```

```
#include "LEDA/edge_map.h"
#include "LEDA/ugraph.h"
#include "../Datenstruktur/CPunkt.h"
#include "../Datenstruktur/CLinie.h"
#include "../Datenstruktur/CWegenetzKnoten.h"
#include "../Datenstruktur/CWegenetzKante.h"
#include <stdlib.h>
```

5.70 CLoesungsgenerator.cpp Dateireferenz

```
#include "CLoesungsgenerator.h"
```

5.71 CLoesungsgenerator.h Dateireferenz

```
#include "CTransportkettengraph.h"
```

Übersicht

- class **CLoesungsgenerator**

5.72 CLPSchnittstelle.cpp Dateireferenz

```
#include "CLPSchnittstelle.h"
#include "CGUI.h"
```

5.73 CLPSchnittstelle.h Dateireferenz

```
#include "LEDA/graph.h"
#include "CSchnittstelleMitGUI.h"
#include "../Datenstruktur/CFPlanGraphKnoten.h"
#include "../Datenstruktur/CFPlanGraphKante.h"
#include "../Datenstruktur/CLinie.h"
#include "../Datenstruktur/CGuetemasse.h"
#include "CDatenbank.h"
```

Übersicht

- class **CLPSchnittstelle**

5.74 CLPSchnittstelleTest.cpp Dateireferenz

```
#include "CLPSchnittstelleTest.h"
```

5.75 CLPSchnittstelleTest.h Dateireferenz

```
#include <iostream.h>
#include "LEDA/graph.h"
#include "CFPlanGraphKnoten.h"
#include "CFPlanGraphKante.h"
```

Übersicht

- class **CLPSchnittstelleTest**

5.76 CNetzplan.cpp Dateireferenz

```
#include "CNetzplan.h"
```

5.77 CNetzplan.h Dateireferenz

```
#include "CNetzplanKnoten.h"
#include "CNetzplanKante.h"
#include "../Datenstruktur/CLinie.h"
#include <LEDA/ugraph.h>
#include <LEDA/list.h>
#include <LEDA/graph_alg.h>
#include "Debug.h"
#include "../Datenstruktur/CWegenetzKnoten.h"
#include "../Datenstruktur/CWegenetzKante.h"
#include "../Datenstruktur/CFPlanGraphKnoten.h"
#include "../Datenstruktur/CFPlanGraphKante.h"
#include "../Datenstruktur/CFortschritt.h"
```

Übersicht

- class **CNetzplan**

5.78 CNetzplanKante.cpp Dateireferenz

```
#include "CNetzplanKante.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CNetzplanKante**& Zielkante)
- ostream& **operator<<** (ostream& os, const **CNetzplanKante**& Ausgabekante)
- int **compare** (const **CNetzplanKante**& kante1, const **CNetzplanKante**& kante2)

5.78.1 Dokumentation der Funktionen

5.78.1.1 int compare (const CNetzplanKante & *kante1*, const CNetzplanKante & *kante2*)

compare Operator

5.78.1.2 ostream & operator<< (ostream & *os*, const CNetzplanKante & *Ausgabekante*)

<< Operator

5.78.1.3 istream & operator>> (istream & *is*, CNetzplanKante & *Zielkante*)

>> Operator

5.79 CNetzplanKante.h Dateireferenz

```
#include <iostream.h>
```

```
#include "Debug.h"
```

Übersicht

- class **CNetzplanKante**

Makrodefinitionen

- #define **CNETZPLANKANTE_H**

5.79.1 Makro-Dokumentation

5.79.1.1 #define CNETZPLANKANTE_H

5.80 CNetzplanKnoten.cpp Dateireferenz

```
#include "CNetzplanKnoten.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CNetzplanKnoten**& zielKante)
- ostream& **operator<<** (ostream& os, const **CNetzplanKnoten**& ausgabeKante)
- int **compare** (const **CNetzplanKnoten**& knoten1, const **CNetzplanKnoten**& knoten2)

5.80.1 Dokumentation der Funktionen

5.80.1.1 int compare (const CNetzplanKnoten & *knoten1*, const CNetzplanKnoten & *knoten2*)

compare Operator

5.80.1.2 ostream & operator<< (ostream & *os*, const CNetzplanKnoten & *ausgabeKante*)

<< Operator

5.80.1.3 istream & operator>> (istream & *is*, CNetzplanKnoten & *zielKante*)

>> Operator

5.81 CNetzplanKnoten.h Dateireferenz

```
#include <iostream.h>
#include <LEDA/edge_map.h>
#include "Debug.h"
```

Übersicht

- class **CNetzplanKnoten**

5.82 CParameterHaltestelle.cpp Dateireferenz

```
#include "Debug.h"
#include "Fehler.h"
#include "CParameterHaltestelle.h"
```

5.83 CParameterHaltestelle.h Dateireferenz

```
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/graph.h>
#include <assert.h>
#include "CPunkt.h"
#include "CBevoelkerungsdichteElement.h"
#include "CQuelleZielListeElement.h"
#include "CPendlerListeElement.h"
```

Übersicht

- class **CParameterHaltestelle**

Aufzählungen

- enum **TAlgorithmus** { **stochastisch**, **stochastischUmgebung**, **evolutionaereStrategie**, **nurPflichthaltestellen** }
- enum **TRadius** { **luftlinie**, **realweg** }
- enum **TAnpassung** { **lot**, **kreuzung** }
- enum **TStrategie** { **plus**, **komma** }

5.83.1 Dokumentation der Aufzählungstypen

5.83.1.1 enum TAlgorithmus

Bestimmt den zu verwendenen Algorithmus fuer die Ermittlung der Haltestellen.

Aufzählungswerte:

stochastisch

stochastischUmgebung

evolutionaereStrategie

nurPflichthaltestellen

5.83.1.2 enum TAnpassung

Beschreibt die Modi der fuer die Anpassung der Haltestellen an das Strassennetz zugrundeliegenden Berechnungsarten.

lot - Das Lot wird von der aktuellen Position zu dem naechstliegenden Strassenzug gefaellt und der Fusspunkt ist die Position der Haltestelle.

kreuzung - Die naechstligende Kreuzung wird die neue Haltestellenposition.

Aufzählungswerte:

lot

kreuzung

5.83.1.3 enum TRadius

Beschreibt die Modi der fuer die Ermittlung von Entfernungen zugrundeliegenden Berechnungsarten.

luftlinie - Die Luftlinie wird als Entfernungsmass zwischen zwei Punkten angenommen.

realweg - Der Realweg wird als Entfernungsmass zwischen zwei Punkten angenommen.

Aufzählungswerte:

luftlinie

realweg

5.83.1.4 enum TStrategie

Dient der Wahl der grundlegenden Strategie der ES.

Aufzählungswerte:

plus

komma

5.84 CPendlerListeElement.cpp Dateireferenz

```
#include "CPendlerListeElement.h"
```

5.85 CPendlerListeElement.h Dateireferenz

```
#include "CPunkt.h"
```

Übersicht

- class **CPendlerListeElement**

5.86 CPlanquadrat.cpp Dateireferenz

```
#include "CPlanquadrat.h"
```

5.87 CPlanquadrat.h Dateireferenz

Übersicht

- class **CPlanquadrat**

5.88 CPunkt.cpp Dateireferenz

```
#include "CPunkt.h"
```

Funktionen

- ostream& **operator<<** (ostream& os, const **CPunkt**& beschriftung)

5.88.1 Dokumentation der Funktionen

5.88.1.1 ostream & operator<< (ostream & *os*, const CPunkt & *beschriftung*)

<< Operator

5.89 CPunkt.h Dateireferenz

```
#include <iostream.h>
#include "Debug.h"
```

Übersicht

- class **CPunkt**

5.90 CQuelleZielListeElement.cpp Dateireferenz

```
#include "CQuelleZielListeElement.h"
```

5.91 CQuelleZielListeElement.h Dateireferenz

```
#include "CPunkt.h"
```

Übersicht

- class **CQuelleZielListeElement**

5.92 CReisebewertung.cpp Dateireferenz

```
#include "CReisebewertung.h"
```

Funktionen

- istream& **operator>>** (istream& instream, **CReisebewertung**& reisebewertung)
- ostream& **operator<<** (ostream& outstream, const **CReisebewertung**& reisebewertung)

5.92.1 Dokumentation der Funktionen

5.92.1.1 ostream & operator<< (ostream & *outstream*, const CReisebewertung & *reisebewertung*)

<< Operator

5.92.1.2 istream & operator>> (istream & *instream*, CReisebewertung & *reisebewertung*)

>> Operator

5.93 CReisebewertung.h Dateireferenz

```
#include <LEDA/list.h>
#include <LEDA/tuple.h>
```

Übersicht

- class **CReisebewertung**

5.94 CSchnittstelle.h Dateireferenz

```
#include "CDatenbank.h"
```

Übersicht

- class **CSchnittstelle**

5.95 CSchnittstelleMitGUI.cpp Dateireferenz

```
#include "CSchnittstelleMitGUI.h"
```

5.96 CSchnittstelleMitGUI.h Dateireferenz

```
#include "CSchnittstelle.h"
#include <iostream.h>
```

Übersicht

- class **CSchnittstelleMitGUI**

5.97 CSchnittstellenTest.cpp Dateireferenz

```
#include "CSchnittstellenTest.h"
```

5.98 CSchnittstellenTest.h Dateireferenz

```
#include "CTransportkettengraph.h"
#include <LEDA/graph.h>
#include "CFPlanGraphKante.h"
#include "CFPlanGraphKnoten.h"
#include "CFahrplan.h"
#include <LEDA/list.h>
#include <LEDA/array.h>
#include <LEDA/tuple.h>
#include <LEDA/h_array.h>
#include "CFahrplanElement.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CSchnittstellenTest**

5.99 CSimplexTableau.cpp Dateireferenz

```
#include "CSimplexTableau.h"
```

Funktionen

- int **VergleichenTupel** (two_tuple<int, int> * const & p, two_tuple<int, int> * const & q)

5.99.1 Dokumentation der Funktionen

5.99.1.1 int VergleichenTupel (two_tuple< int,int >*const & *p*, two_tuple< int,int >*const & *q*)

5.100 CSimplexTableau.h Dateireferenz

```
#include "CKantenBeschriftung.h"
#include "CKnotenBeschriftung.h"
#include "LEDA/stack.h"
#include "LEDA/graph.h"
#include "LEDA/matrix.h"
#include "LEDA/vector.h"
#include "LEDA/list.h"
#include "LEDA/array.h"
#include "LEDA/integer.h"
#include "Debug.h"
#include "LEDA/tuple.h"
#include "CSpMatrix.h"
#include "CFortschritt.h"
```

Übersicht

- class **CSimplexTableau**

5.101 CSpMatrix.cpp Dateireferenz

```
#include <stdio.h>
#include "CSpMatrix.h"
```

Makrodefinitionen

- #define **VM_BOUND_CHECK**

Funktionen

- ostream& **operator**<< (ostream & os, const **TSparseMatrix** & M)

5.101.1 Makro-Dokumentation

5.101.1.1 #define VM_BOUND_CHECK

5.101.2 Dokumentation der Funktionen

5.101.2.1 ostream & operator<< (ostream & *os*, const TSparse-Matrix & *M*)

5.102 CSpMatrix.h Dateireferenz

```
#include <iostream.h>
```

```
#include "CSpVektor.h"
```

Übersicht

- struct **tagTRow**
- class **TSparseMatrix**

Makrodefinitionen

- #define **UINT** unsigned int

Typendefinitionen

- typedef struct **tagTRow TRow**

5.102.1 Makro-Dokumentation

5.102.1.1 #define UINT unsigned int

5.102.2 Dokumentation der benutzerdefinerten Typen

5.102.2.1 typedef struct tagTRow TRow

5.103 CSpVektor.cpp Dateireferenz

```
#include "stdlib.h"
```

```
#include "CSpVektor.h"
```

Funktionen

- **TSparseVector operator** ∗ (const double d, **TSparseVector** V)
- **TSparseVector operator** ∗ (**TSparseVector** V, const double d)
- ostream& **operator<<** (ostream & os, const **TSparseVector** & v)

5.103.1 Dokumentation der Funktionen

5.103.1.1 TSparseVector operator ∗ (TSparseVector *V*, const double *d*)

5.103.1.2 TSparseVector operator ∗ (const double *d*, TSparseVector *V*)

5.103.1.3 ostream & operator<< (ostream & *os*, const TSparseVector & *v*)

5.104 CSpVektor.h Dateireferenz

```
#include <iostream.h>
```

```
#include "CSpMatrix.h"
```

Übersicht

- struct **tagTEntry**
- class **TSparseVector**

Makrodefinitionen

- #define **UINT** unsigned int

Typendefinitionen

- typedef struct **tagTEntry TEntry**

5.104.1 Makro-Dokumentation

5.104.1.1 #define UINT unsigned int

5.104.2 Dokumentation der benutzerdefinerten Typen

5.104.2.1 typedef struct tagTEntry TEntry

5.105 CStartverknuepfung.cpp Dateireferenz

```
#include "CStartverknuepfung.h"
```

5.106 CStartverknuepfung.h Dateireferenz

```
#include "CNetzplanKnoten.h"
#include "CNetzplanKante.h"
#include "CLinie.h"
#include "CNetzplan.h"
#include "CBasislinie.h"
#include "Debug.h"
```

Übersicht

- class **CStartverknuepfung**

5.107 CStrassennetz.cpp Dateireferenz

```
#include <math.h>
#include <time.h>
#include "CStrassennetz.h"
```

Makrodefinitionen

- #define **DEBUG_berechnenKreuzungen** 10
- #define **DEBUG_berechnenSchnittpunkt** 10
- #define **DEBUG_berechnenPunkt** 10
- #define **DEBUG_Verkehrsbedarfsmatrix** 10
- #define **DEBUG_erstellenKantenListe** 10

Funktionen

- int **cmpVerbindungsListe** (**CVerbindung** * const & V1, **CVerbindung** * const & V2)

5.107.1 Makro-Dokumentation

5.107.1.1 #define DEBUG_Verkehrsbedarfsmatrix 10

5.107.1.2 #define DEBUG_berechnenKreuzungen 10

5.107.1.3 #define DEBUG_berechnenPunkt 10

5.107.1.4 #define DEBUG_berechnenSchnittpunkt 10

5.107.1.5 #define DEBUG_erstellenKantenListe 10

5.107.2 Dokumentation der Funktionen

5.107.2.1 int cmpVerbindungsListe (CVerbindung *const & *V1*, CVerbindung *const & *V2*)

5.108 CStrassennetz.h Dateireferenz

```
#include <LEDA/graph.h>
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/tuple.h>
#include <LEDA/rat_geo_alg.h>
#include <LEDA/graph_misc.h>
#include "../Datenstruktur/CWegenetzKnoten.h"
#include "../Datenstruktur/CWegenetzKante.h"
#include "../Datenstruktur/CPunkt.h"
#include "../Programm/CHSSchnittstelle.h"
#include "../Datenstruktur/CParameterHaltestelle.h"
#include "../Datenstruktur/CGuetemasse.h"
#include "../Datenstruktur/Debug.h"
#include "../Datenstruktur/define.h"
#include "CVisualisierung.h"
#include "CFortschritt.h"
#include "CVerbindung.h"
```

Übersicht

- class **CStrassennetz**

5.109 CStrassennetzES.cpp Dateireferenz

```
#include "CStrassennetzES.h"
```

5.110 CStrassennetzES.h Dateireferenz

```
#include <LEDA/graph.h>
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/tuple.h>
#include "CStrassennetz.h"
#include "./../Datenstruktur/CPunkt.h"
#include "./../Datenstruktur/CWegenetzKnoten.h"
#include "teaCESVector.h"
#include "teaISimple.h"
#include "teaPES.h"
#include "teaOperator.h"
#include "teaLottery.h"
```

Übersicht

- class **CStrassennetzES**

5.111 CStrassennetzGraph.cpp Dateireferenz

```
#include "CStrassennetzGraph.h"
```

Funktionen

- list<CStrasse *>* **ermittelnStrassenAusPlanquadraten** (Planquadrate)

5.111.1 Dokumentation der Funktionen

5.111.1.1 list< CStrasse *>* ermittelnStrassenAusPlanquadraten<CStrasse *> (Planquadrate)

5.112 CStrassennetzGraph.h Dateireferenz

Übersicht

- class **CStrassennetzGraph**

5.113 CStrassennetzStochastisch.cpp Dateireferenz

```
#include "CStrassennetzStochastisch.h"
```

5.114 CStrassennetzStochastisch.h Dateireferenz

```
#include <LEDA/graph.h>
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/tuple.h>
#include "CStrassennetz.h"
#include "./../Datenstruktur/CPunkt.h"
#include "./../Datenstruktur/CWegenetzKnoten.h"
```

Übersicht

- class **CStrassennetzStochastisch**

5.115 CStrassennetzStochastischUmgebung.cpp Dateireferenz

```
#include "CStrassennetzStochastischUmgebung.h"
```

5.116 CStrassennetzStochastischUmgebung.h Dateireferenz

```
#include <LEDA/graph.h>
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/tuple.h>
#include "CStrassennetz.h"
#include "./../Datenstruktur/CPunkt.h"
#include "./../Datenstruktur/CWegenetzKnoten.h"
```

Übersicht

- class **CStrassennetzStochastischUmgebung**

5.117 CSukzessivVerfahren.cpp Dateireferenz

```
#include "CSukzessivVerfahren.h"
```

5.118 CSukzessivVerfahren.h Dateireferenz

```
#include "CLoesungsgenerator.h"
#include <LEDA/graph.h>
```

Übersicht

- class **CSukzessivVerfahren**

Makrodefinitionen

- #define **SUKZESSIVVERFAHREN_H**

5.118.1 Makro-Dokumentation

5.118.1.1 #define SUKZESSIVVERFAHREN_H

5.119 CTestklasse0.cpp Dateireferenz

```
#include "CTestklasse0.h"
#include "Testkoordinaten.h"
#include "../Datenstruktur/Debug.h"
#include <stdlib.h>
```

5.120 CTestklasse0.h Dateireferenz

```
#include "CDatenbank.h"
```

Übersicht

- class **CTestklasse0**

5.121 CTestklasse1.cpp Dateireferenz

```
#include "CTestklasse1.h"
#include "Testkoordinaten.h"
#include "../Datenstruktur/Debug.h"
#include <stdlib.h>
```

5.122 CTestklasse1.h Dateireferenz

```
#include "CDatenbank.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CTestklasse1**

5.123 CTestklasse2.cpp Dateireferenz

```
#include "CTestklasse2.h"
#include "Testkoordinaten.h"
#include "../Datenstruktur/Debug.h"
#include <stdlib.h>
```

5.124 CTestklasse2.h Dateireferenz

```
#include "CDatenbank.h"
```

Übersicht

- class **CTestklasse2**

5.125 CTestklasse3.cpp Dateireferenz

```
#include "CTestklasse3.h"
#include "Testkoordinaten.h"
#include "../Datenstruktur/Debug.h"
#include <stdlib.h>
```

5.126 CTestklasse3.h Dateireferenz

```
#include "CDatenbank.h"
```

Übersicht

- class **CTestklasse3**

5.127 CTestklasse4.cpp Dateireferenz

```
#include "CTestklasse4.h"
```

5.128 CTestklasse4.h Dateireferenz

```
#include "CDatenbank.h"
#include "LEDA/segment.h"
#include "LEDA/point.h"
#include "Debug.h"
```

Übersicht

- class **CTestklasse4**

5.129 CTestklasse5.cpp Dateireferenz

```
#include "CTestklasse5.h"
```

5.130 CTestklasse5.h Dateireferenz

```
#include "CDatenbank.h"
```

Übersicht

- class **CTestklasse5**

5.131 CTestklasse6.cpp Dateireferenz

```
#include "CTestklasse6.h"
#include "Testkoordinaten.h"
#include "../Datenstruktur/Debug.h"
#include <stdlib.h>
```

5.132 CTestklasse6.h Dateireferenz

```
#include "CDatenbank.h"
#include "CGuetemasse.h"
```

Übersicht

- class **CTestklasse6**

5.133 CTransportkettengraph.cpp Dateireferenz

```
#include "CTransportkettengraph.h"
```

5.134 CTransportkettengraph.h Dateireferenz

```
#include "CZusammenhangskomponenten.h"
#include "CKantenBeschriftung.h"
#include "CKnotenBeschriftung.h"
#include <iostream.h>
#include "LEDA/graph.h"
#include "LEDA/h_array.h"
#include "LEDA/list.h"
#include "LEDA/basic.h"
#include "LEDA/array.h"
#include "LEDA/tuple.h"
#include "LEDA/map2.h"
#include "CFPlanGraphKante.h"
#include "CFPlanGraphKnoten.h"
#include "CFahrplanElement.h"
#include "LEDA/graph_alg.h"
#include "LEDA/node_array.h"
#include "Debug.h"
#include "CVisualFahrplan.h"
#include "CReisebewertung.h"
```

Übersicht

- class **CTransportkettengraph**

5.135 CVerbindung.cpp Dateireferenz

```
#include "CVerbindung.h"
```

5.136 CVerbindung.h Dateireferenz

```
#include <LEDA/graph.h>
#include "../Datenstruktur/CPunkt.h"
```

Übersicht

- class **CVerbindung**

5.137 CVergleichsklasse.cpp Dateireferenz

```
#include "CVergleichsklasse.h"
```

5.138 CVergleichsklasse.h Dateireferenz

```
#include "LEDA/graph.h"
#include "CKantenBeschriftung.h"
#include "CKnotenBeschriftung.h"
#include <iostream.h>
```

Übersicht

- class **vergleichenKanten**

5.139 CVerknuepfungsphase.cpp Dateireferenz

```
#include "CVerknuepfungsphase.h"
```

5.140 CVerknuepfungsphase.h Dateireferenz

```
#include "CAbschlussverknuepfung.h"
#include "CStartverknuepfung.h"
#include "CBasislinie.h"
#include "CNetzplan.h"
#include "Debug.h"
#include "LEDA/list.h"
#include "../Programm/CLPSchnittstelle.h"
```

Übersicht

- class **CVerknuepfungsphase**

5.141 CVisualFahrplan.cpp Dateireferenz

```
#include "CVisualFahrplan.h"
```

5.142 CVisualFahrplan.h Dateireferenz

```
#include "LEDA/graph.h"
#include "LEDA/graph_alg.h"
#include "LEDA/graphwin.h"
```

Übersicht

- class **CVisualFahrplan**

5.143 CVisualisierung.cpp Dateireferenz

```
#include "CVisualisierung.h"
```

5.144 CVisualisierung.h Dateireferenz

```
#include <LEDA/window.h>
#include <LEDA/matrix.h>
#include <LEDA/graph.h>
#include <LEDA/rat_point.h>
#include <LEDA/rat_segment.h>
#include "Debug.h"
#include "../Datenstruktur/CWegenetzKnoten.h"
```

Übersicht

- class **CVisualisierung**

5.145 CWegenetzKante.cpp Dateireferenz

```
#include "CWegenetzKante.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CWegenetzKante**& beschriftung)
- ostream& **operator<<** (ostream& os, const **CWegenetzKante**& beschriftung)

5.145.1 Dokumentation der Funktionen

5.145.1.1 ostream & operator<< (ostream & *os*, const CWegenetzKante & *beschriftung*)

<< Operator

5.145.1.2 istream & operator>> (istream & *is*, CWegenetzKante & *beschriftung*)

>> Operator

5.146 CWegenetzKante.h Dateireferenz

```
#include <iostream.h>
#include "Debug.h"
#include "CPunkt.h"
#include <LEDA/graph.h>
#include <LEDA/list.h>
#include <LEDA/matrix.h>
#include <LEDA/tuple.h>
```

Übersicht

- class **CWegenetzKante**

5.147 CWegenetzKnoten.cpp Dateireferenz

```
#include "CWegenetzKnoten.h"
```

Funktionen

- istream& **operator>>** (istream& is, **CWegenetzKnoten**& beschriftung)
- ostream& **operator<<** (ostream& os, const **CWegenetzKnoten**& beschriftung)

5.147.1 Dokumentation der Funktionen

5.147.1.1 ostream & operator<< (ostream & *os*, const CWegenetzKnoten & *beschriftung*)

<< Operator

5.147.1.2 istream & operator>> (istream & *is*, CWegenetzKnoten & *beschriftung*)

>> Operator

5.148 CWegenetzKnoten.h Dateireferenz

```
#include <iostream.h>
#include "CPunkt.h"
#include "Debug.h"
```

Übersicht

- class **CWegenetzKnoten**

5.149 CZusammenhangskomponenten.cpp Dateireferenz

```
#include "CZusammenhangskomponenten.h"
```

Funktionen

- int **sortierenNachLinie** (**CFahrplanElement**∗ const & eintrag1, **CFahrplanElement**∗ const & eintrag2)
- int **sortierenNachFahrzeit** (**CFahrplanElement**∗ const & eintrag1, **CFahrplanElement**∗ const & eintrag2)

5.149.1 Dokumentation der Funktionen

5.149.1.1 int sortierenNachFahrzeit (CFahrplanElement ∗const & *eintrag1*, CFahrplanElement ∗const & *eintrag2*)

5.149.1.2 int sortierenNachLinie (CFahrplanElement ∗const & *eintrag1*, CFahrplanElement ∗const & *eintrag2*)

Diese Funktion ist die Vergleichsfunktion, welche zur Sortierung der berechneten Fahrplan liste benoetigt wird. Diese Sortierung ist Bestandteil der Fahrzeugumlaufplannung.

5.150 CZusammenhangskomponenten.h Dateireferenz

```
#include "CKantenBeschriftung.h"
#include "CKnotenBeschriftung.h"
#include "CSimplexTableau.h"
#include "CFahrplanElement.h"
```

```
#include <iostream.h>
#include "LEDA/graph.h"
#include "LEDA/graph_alg.h"
#include "LEDA/tuple.h"
#include "LEDA/h_array.h"
#include "LEDA/list.h"
#include "LEDA/node_partition.h"
#include "LEDA/string.h"
#include "LEDA/map2.h"
#include "CVergleichsklasse.h"
#include <math.h>
#include "Debug.h"
#include "Fehler.h"
#include "CVisualFahrplan.h"
#include "CReisebewertung.h"
#include "LEDA/templates/shortest_path.t"
#include "CFortschritt.h"
```

Übersicht

- class **CZusammenhangskomponenten**

5.151 Debug.cpp Dateireferenz

```
#include "Debug.h"
```

Funktionen

- void **wartenTaste** (void)

5.151.1 Dokumentation der Funktionen

5.151.1.1 void wartenTaste (void)

Gibt eine kurze Meldung aus und wartet auf das Druecken einer Taste.

5.152 Debug.h Dateireferenz

```
#include "../Datenstruktur/Fehler.h"
```

```
#include <iostream.h>

#include "define.h"
```

Makrodefinitionen

- #define **DEBUG_KEINE_MELDUNG** 0
- #define **DEBUG_OBJEKT** 1
- #define **DEBUG_METHODE** 2
- #define **DEBUG_METHODE_FUNDAMENTAL** 2
- #define **DEBUG_METHODE_HILFE** 3
- #define **DEBUG_METHODE_SUB** 4
- #define **DEBUG_ALGO** 5
- #define **DEBUG_ALGORITHMUS** 5
- #define **DEBUG_ALGORITHMUS_DETAIL** 6
- #define **DEBUG_RESERVIERT** 7
- #define **DEBUG_ALGORITHMEN_SUB** 8
- #define **DEBUG_ALLES** 9
- #define **Debug**(a,b) if(a<=DEBUG_LEVEL){cerr<<"[DEBUG:"<<__FILE__<<":"<<__LINE__<<"] ";if(a==1)cerr<<"[Objekt] ";if(a==2)cerr<<"[MetFund] ";if(a==3)cerr<<"[MetHilf] ";if(a==4)cerr<<"[MetRest] ";if(a==5)cerr<<"[AlgoStE] ";if(a==6)cerr<<"[AlgoDet] ";if(a==7)cerr<<"[reservd] ";if(a==8)cerr<<"[AlgoSub] ";if(a==9)cerr<<"[rest] ";cerr<<b<<endl;cerr.flush();}
- #define **Error**(a) cerr << "Fehler: " << a << "\n"; cerr.flush();

5.152.1 Makro-Dokumentation

5.152.1.1 #define DEBUG_ALGO 5

5.152.1.2 #define DEBUG_ALGORITHMEN_SUB 8

5.152.1.3 #define DEBUG_ALGORITHMUS 5

5.152.1.4 #define DEBUG_ALGORITHMUS_DETAIL 6

5.152.1.5 #define DEBUG_ALLES 9

5.152.1.6 #define DEBUG_KEINE_MELDUNG 0

Dient dem Ausgeben von Debuginformationen.

Die Level sind hierbei wie folgt definiert:

...

Parameter:

level Level, bis dem die Meldung ausgegeben werden soll.

debugtext Auszugebende Meldung.

5.152.1.7 #define DEBUG_METHODE 2

5.152.1.8 #define DEBUG_METHODE_FUNDAMENTAL 2

5.152.1.9 #define DEBUG_METHODE_HILFE 3

5.152.1.10 #define DEBUG_METHODE_SUB 4

5.152.1.11 #define DEBUG_OBJEKT 1

5.152.1.12 #define DEBUG_RESERVIERT 7

5.152.1.13 #define Debug(a, b) if(a<=DEBUG_LEVEL){cerr<<"[DEBUG:"<<__FILE__<<":"<<__LINE__<<"] ";if(a==1)cerr<<"[Objekt] ";if(a==2)cerr<<"[MetFund] ";if(a==3)cerr<<"[MetHilf] ";if(a==4)cerr<<"[MetRest] ";if(a==5)cerr<<"[AlgoStE] ";if(a==6)cerr<<"[AlgoDet] ";if(a==7)cerr<<"[reservd] ";if(a==8)cerr<<"[AlgoSub] ";if(a==9)cerr<<"[rest] ";cerr<<b<<endl;cerr.flush();}

5.152.1.14 #define Error(a) cerr << "Fehler: " << a << "\n"; cerr.flush();

5.153 define.h Dateireferenz

Makrodefinitionen

- #define **DEBUG_LEVEL** 9
- #define **MIT_GRASS**

Variablen

- const double **MAX_DOUBLE** = 10000000.0
- const double **MIN_DOUBLE** = -10000000.0
- const double **PI** = 3.4142

5.153.1 Makro-Dokumentation

5.153.1.1 #define DEBUG_LEVEL 9

5.153.1.2 #define MIT_GRASS

5.153.2 Variablen-Dokumentation

5.153.2.1 const double MAX_DOUBLE = 10000000.0

5.153.2.2 const double MIN_DOUBLE = -10000000.0

5.153.2.3 const double PI = 3.4142

5.154 Fehler.cpp Dateireferenz

```
#include "Fehler.h"
```

5.155 Fehler.h Dateireferenz

```
#include "LEDA/string.h"
```

Übersicht

- class **AusserhalbWertebereich**
- class **CFahrplanExceptions**
- class **CFesteBindungNichtGefunden**
- class **CGuetemasseFehler**
- class **CHaltestelleExceptions**
- class **CKreisGewaehlt**
- class **CUnsinnigerWertGeliefert**
- class **NichtInstanzierteParameter**
- class **UnsinnigeErzeugung**
- class **WohldefinierterAbbruch**

5.156 Haltestelle.cpp Dateireferenz

```
#include "Haltestelle.h"
#include "CFortschritt.h"
```

Funktionen

- int **main** (int argc, char ** argv)

5.156.1 Dokumentation der Funktionen

5.156.1.1 int main (int *argc*, char ** *argv*)

4 Punkte, die die Geraden definieren: 1. Strasse (x=2,y=15) (x=5,y=33) 2. Strasse (x=1.5,y=0) (x=7.35,y=18.72) Schnittpunkt: x=-2.79, y=-13.74

5.157 Haltestelle.h Dateireferenz

```
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <sys/types.h>
#include <time.h>
#include "CHaltestelle.h"
#include "CPunkt.h"
#include "CStrassennetz.h"
```

Übersicht

- class **CTestStrassennetz**

Makrodefinitionen

- #define **VERSION** "V0.01"

5.157.1 Makro-Dokumentation

5.157.1.1 #define VERSION "V0.01"

5.158 iomanip.h Dateireferenz

5.159 linienplandummydaten.cpp Dateireferenz

```
#include "linienplandummydaten.h"
```

Funktionen

- void **linienplandummydaten** (**CLPSchnittstelle*** programmSchnittstelle, **CLPSchnittstelleTest*** programmSchnittstelleTest)

5.159.1 Dokumentation der Funktionen

5.159.1.1 void linienplandummydaten (CLPSchnittstelle * *programmSchnittstelle*, CLPSchnittstelleTest * *programmSchnittstelleTest*)

5.160 linienplandummydaten.h Dateireferenz

Funktionen

- void **linienplandummydaten** (**CLPSchnittstelle*** programmSchnittstelle)

5.160.1 Dokumentation der Funktionen

5.160.1.1 void linienplandummydaten (CLPSchnittstelle * *programmSchnittstelle*)

5.161 Programm.cpp Dateireferenz

```
#include "CGUI.h"
#include "CKontrolle.h"
#include "CGUISchnittstelle.h"
#include "CGRASSSchnittstelle.h"
#include <iostream>
#include <unistd.h>
#include <cstdlib>
#include <limits.h>
#include "LEDA/graph.h"
#include "LEDA/list.h"
```

5.162 Programm.h Dateireferenz

5.163 Testdaten.cpp Dateireferenz

5.164 Testdaten.h Dateireferenz

5.165 Testkoordinaten.h Dateireferenz

Makrodefinitionen

- #define **P01** 2577000,5717000

- #define **P02** 2577050,5716850
- #define **P03** 2577000,5716700
- #define **P04** 2577150,5716750
- #define **P05** 2577200,5716500
- #define **P06** 2577300,5716200
- #define **P07** 2577400,5716050
- #define **P08** 2577450,5716000
- #define **P09** 2577450,5716250
- #define **P10** 2577500,5716400
- #define **P11** 2577600,5716350
- #define **P12** 2577500,5716800
- #define **P13** 2577500,5717150
- #define **P14** 2577550,5717350
- #define **P15** 2577600,5717100
- #define **P16** 2577600,5716950
- #define **P17** 2577750,5717000
- #define **P18** 2577750,5716800
- #define **P19** 2577850,5716600
- #define **P20** 2577900,5716600
- #define **P21** 2577950,5716600
- #define **P22** 2577900,5716450
- #define **P23** 2578000,5716100
- #define **P24** 2578250,5716150
- #define **P25** 2578500,5716250
- #define **P26** 2577950,5716800
- #define **P27** 2578000,5717000
- #define **P28** 2578200,5716800
- #define **P29** 2578400,5716950
- #define **P30** 2578400,5717150
- #define **P31** 2578300,5717250
- #define **P32** 2578450,5717200
- #define **P33** 2578850,5716800
- #define **P34** 2578200,5716400
- #define **P35** 2579200,5716450

5.165.1 Makro-Dokumentation

5.165.1.1 #define P01 2577000,5717000

5.165.1.2 #define P02 2577050,5716850

5.165.1.3 #define P03 2577000,5716700

5.165.1.4 #define P04 2577150,5716750

5.165.1.5 #define P05 2577200,5716500

5.165.1.6 #define P06 2577300,5716200

5.165.1.7 #define P07 2577400,5716050

5.165.1.8 #define P08 2577450,5716000

5.165.1.9 #define P09 2577450,5716250

5.165.1.10 #define P10 2577500,5716400

5.165.1.11 #define P11 2577600,5716350

5.165.1.12 #define P12 2577500,5716800

5.165.1.13 #define P13 2577500,5717150

5.165.1.14 #define P14 2577550,5717350

5.165.1.15 #define P15 2577600,5717100

5.165.1.16 #define P16 2577600,5716950

5.165.1.17 #define P17 2577750,5717000

5.165.1.18 #define P18 2577750,5716800

5.165.1.19 #define P19 2577850,5716600

5.165.1.20 #define P20 2577900,5716600

5.165.1.21 #define P21 2577950,5716600

5.165.1.22 #define P22 2577900,5716450

5.165.1.23 #define P23 2578000,5716100

5.165.1.24 #define P24 2578250,5716150

5.165.1.25 #define P25 2578500,5716250

5.165.1.26 #define P26 2577950,5716800

5.165.1.27 #define P27 2578000,5717000

5.165.1.28 #define P28 2578200,5716800

5.165.1.29 #define P29 2578400,5716950

5.165.1.30 #define P30 2578400,5717150

5.165.1.31 #define P31 2578300,5717250

5.165.1.32 #define P32 2578450,5717200

5.165.1.33 #define P33 2578850,5716800

5.165.1.34 #define P34 2578200,5716400

5.165.1.35 #define P35 2579200,5716450

Index