Generikus irányítatlan gráf

Készítette Doxygen 1.8.13

Tartalomjegyzék

1. fejezet

Generikus irányítatlan gráf specifikáció

1.1. Generikus irányítatlan gráf specifikáció

1.1.1. Feladatkiírás

Készítsen generikus irányítatlan gráfot! A gráf megadása szomszédsági mátrixszal történjen! A csomópontokat osztállyal reprezentálja! Definiáljon műveleteket annak meghatározására, hogy a gráfnak hány csomópontja és hány éle van! Szélességi bejárással állapítsa meg, hogy a gráf összefüggő-e! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz **ne** használjon STL tárolót!

1.1.2. Bevezetés

A program elsődleges célja generikus irányítatlan gráfok tárolása és ezekkel történő műveletek végzése. Ezen felül nyújtson segítséget a BSZ2 tárgynak a feladatkiírásban szereplő részeinek a mélyebb megértésében és az ehhez kapcsolódó feladatok ellenőrzésében.

1.1.3. A program képességei

A gráf kifejezés alatt mindenhol egy n csúcsú és e élű irányítatlan egyszerű G gráfot értek.

- Gráfok eltárolása a programban
  - Szomszédsági mátrix beolvasása fájlból.
  - Szomszédsági mátrix kiírása fájlba.
  - Szomszédsági mátrix megjelenítése a standard outputon.
- Műveletek gráfokkal
  - Csomópont szám meghatározása
  - Élszám meghatározása
  - Gráf összefüggőségének vizsgálata
- Tesztprogram
  - A Programhoz tartozik tesztprogram is, ami a fenti képességek helyességét ellenőrzi.

1.1.4. Gráfok eltárolása

A program legyen képes olyan módon gráfokat tárolni, amely lehetővé teszi a műveletvégzést.

1.1.4.1. Gráfok beolvasása

A gráf megadása egy M szomszédsági mátrixszal történik. Az M mátrix n x n-es méretű. Az M mátrix $a_{1,2}$

eleme 1, ha a 1-es és 2-es indexű csúcsok között él húzódik és 0, ha nem. A csúcsok tetszőleges típusú adatokat tartalmazhatnak.

Szomszédsági mátrix minta:

```
n
a1, a2, a3, a4 ... an
b1, b2, b3, b4 ... bn
.   .   .   .       .
.   .   .   .       .
.   .   .   .       .
n1, n2, n3, n4 ... nn
```

Minden T típusú adathoz létezik egy függvény, ami elvégzi annak az adatnak a beolvasását. Ennek az elkészítése és a helyesség biztosítása a felhasználó dolga.

1.1.4.2. Gráfok kiírása

A gráf kiírása a bemenethez hasonló szomszédsági mátrixszal történik.

1.1.5. Gráf összefüggőségének vizsgálata

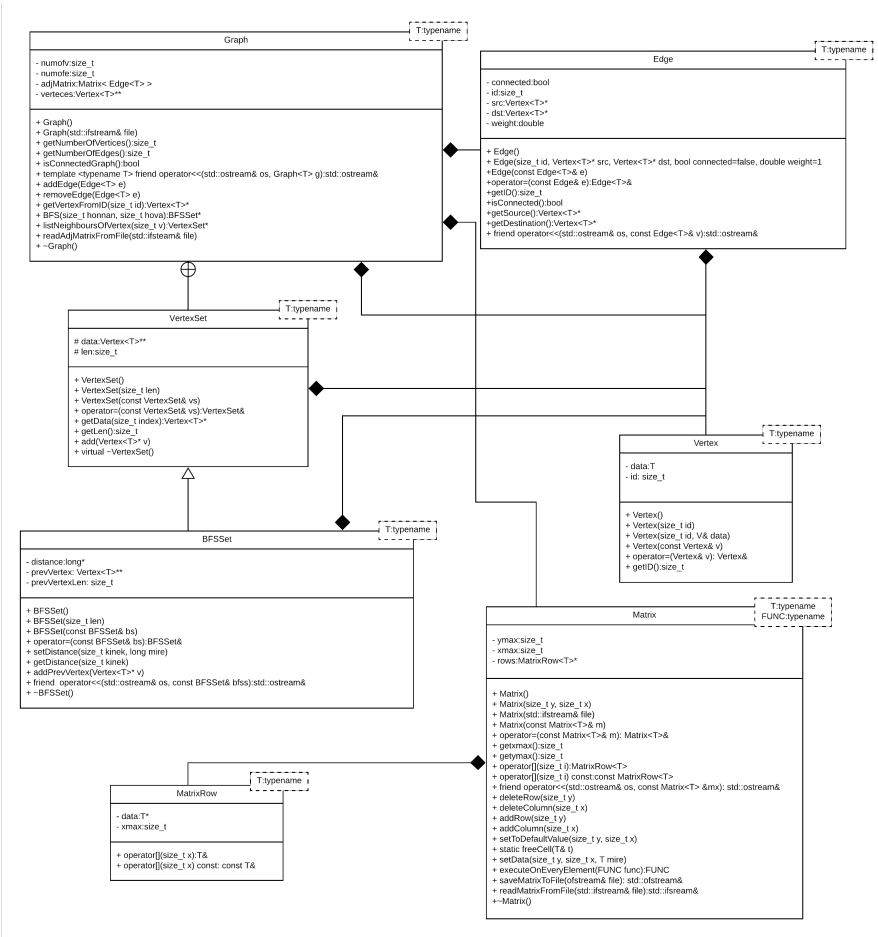
A feladat megvalósításához a program a BSZ2 tárgy során tanult BFS algoritmust fogja használni.

2. fejezet

Generikus irányítatlan gráf terv

2.1. Generikus irányítatlan gráf terv

2.1.1. Osztályok



A feladat megvalósításához előreláthatólag 7 db osztályt használok fel. A tanult eszközök közül az öröklést, az egymásba ágyazott osztályokat és a tartalmazást tervezem felhasználni. Az ábrán még nem szerepel, de a diagram bővülni fog a teszteléshez használt példa osztályokkal.

2.1.1.1. Felhasznált osztályok

2.1.1.1.1. Graph Osztály

Ez az osztály fogja össze a maradék 6 db osztályt. Ez az osztály tartalmazza, a feladatkiírásban szereplő feladatokat megvalósító függvényeket. A legtöbb feladat több segédfüggvényre van bontva.

2.1.1.1.2. Edge Osztály

Ez az osztály foglalja össze az élekhez tartozó információkat. Az osztály megkülönbözteti az élnek a forrás és cél csúcsát. Ezzel a felkészítve arra, hogy a jövőben irányított gráfokat is tudjon kezelni. A `weight` változó lehetővé teszi, hogy az éleket súlyokkal lássuk el, ez szintén a jövőbeli továbbfejlesztés lehetőségét szolgálja.

2.1.1.1.3. Vertex Osztály

Ez az osztály tárolja a csúcsokhoz tartozó adatokat. Mindent csúcsot ellátok egy azonosítóval. Ez a BFS algoritmus során segít a csúcsok megkülönböztetésében és így tudok valamilyen módon hivatkozni a csúcsokra.

2.1.1.1.4. VertexSet Osztály

Ezt az osztályt azért hoztam létre, hogy a függvényekből visszatérő adatokat egységbe tudjam foglalni. Így A dinamikus tömb és annak mérete egymás mellett tárolódik, könnyen elérhető. Az osztály a `Graph` osztály részét képezi, mert logikailag oda tartozik.

2.1.1.1.5. BFFSet Osztály

Ez az osztály a `VertexSet` osztályból származik. A célja hasonló. Ez kifejezetten a BFS algoritmus visszatérési értékének a tárolója.

2.1.1.1.6. Matrix Osztály

Ez az általános felhasználásra szánt osztályt felelős a fájlból történő beolvasás mátrixának tárolására. Az osztály tervezése során felhasználtam a programozás 1 tárgy keretein belül készített házim(mátrix függvénykönyvtár) létrehozása során szerzett tapasztalataimat.

2.1.1.1.7. MatrixRow Osztály

Ez az osztály azért jött létre, hogy a mátrix osztály kettős indexelését lehetővé tegye. Ezen kívül ennek az osztálynak a segítségével elkerülhető, hogy a Mátrix osztályban `T***` típusú adatot kelljen tárolni.

2.1.1.2. Bővíthetőség

A program tervezése során lehetőséget biztosítottam a jövőbeli bővíthetőségre és továbbfejlesztésre.

- A mátrix és a gráf generikus, így bármilyen adatot képes tárolni
- Az élek fel vannak készítve irányított gráfok által való használatra
- Az élekben található `weight` paraméter lehetőséget teremt, hogy a gráf éleit súlyokkal lássuk el.

2.1.2. Felhasznált algoritmus

A specifikációban már említettem, hogy a program célja a BSZ2 tárgyból tanultak elmélyítése. Így a szélességi bejárás algoritmusát is a tárgyban tanultak alapján szeretném megvalósítani. Az algoritmus és annak teljes leírása itt érhető el. Szeretném kiemelni az alábbi fontosabb részt:

- $b(i)$ $(i = 1,2,\dots)$: az i-edikként **b**ejárt csúcs
- $t(v)$ $(v \in V)$: v **t**ávolsága s-től
- $m(v)$ $(v \in V, v \neq s)$: a v-t **m**egelőző csúcs az algoritmus által megtalált, s-ből v-be vezető legrövidebb úton
- j: az eddig bejárt csúcsok száma
- k: a jelenleg aktív csúcs sorszáma a $b(1), b(2), \dots$ sorozatban

BFS ALGORITMUS

Bemenet: Egy $G = (V,E)$ gráf és egy $s \in V$ csúcs

```
1   j ← 1; k ← 1; b(1) ← s
2   t(s) ← 0; minden v ∈ V, v ≠ s-re t(v) ← ∗
3   ciklus
4       ha a b(k) csúcsnak van olyan v szomszédja, amelyre t(v) = ∗, akkor:
5           j ← j + 1
6           b(j) ← v
7           t(v) ← t(b(k)) + 1
8           m(v) ← b(k)
9       különben:
10          ha k = j, akkor:
11              stop
12          különben:
13              k ← k + 1
14  ciklus vége
```

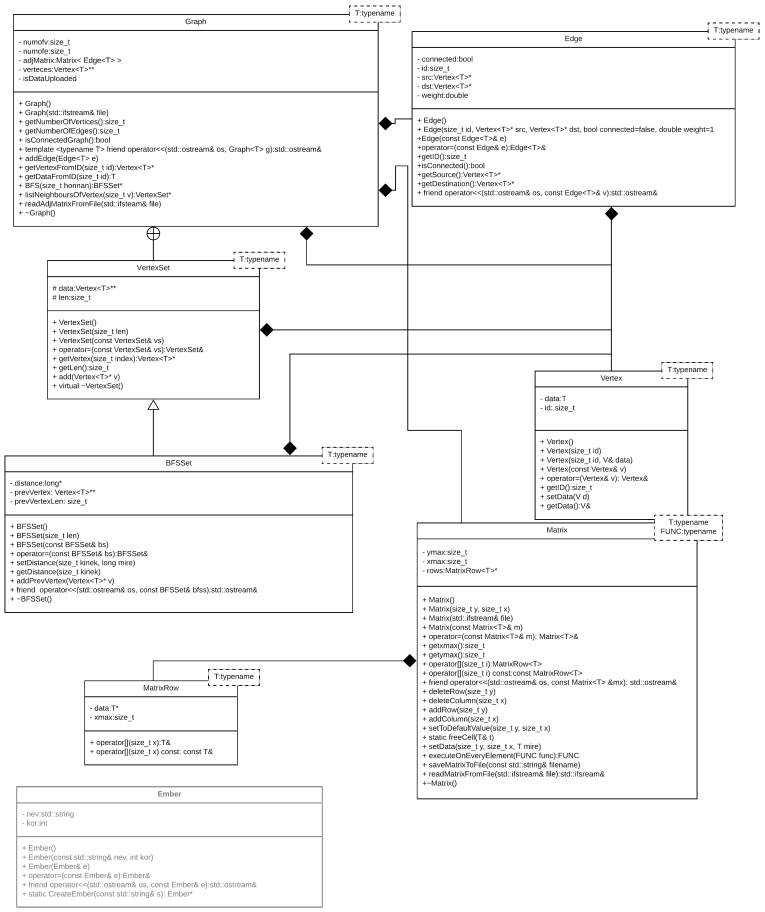
Az algoritmusom megvalósítása közben hasonló változóneveket szeretnék használni. Az algoritmust jelenleg egy helyen tervezem módosítani. A $*$ távolság helyett -1-et tervezek használni. Nálam ez lesz a végtelen jelölése, ez hibakeresés során jól megkülönböztethető lesz a többi távolság értéktől.

3. fejezet

Programozói Dokumentáció

3.1. Programozói Dokumentáció

3.1.1. Véglegesített osztályok



Az elkészített program a fent látható osztályokat tartalmazza. A tervben szereplő UML diagramhoz képest néhány kisebb és 1 darab nagyobb változtatást tartalmaz. Ezen az ábrán már megtalálható egy Ember osztály is. Ez az osztályt a teszteléshez készítettem, ezért nem szerepelt még a tervben szereplő UML diagramban.

3.1.2. A program részei

Részletes leírás a dokumentáció második felében található.

3.1.2.1. graph.hpp

Ebben a fájlban található a gráfhoz és az azon végzek feladatok megvalósításához szükséges osztályok (`Graph` , `Graph< T >::VertexSet`, `Graph< T >::BFSSet`) és függvények.

3.1.2.2. vertex.hpp

Ebben a fájlban található a csúcsokhoz tartozó (`Vertex`) osztály és annak függvényei.

3.1.2.3. edge.hpp

Ebben a fájlban található az élekhez tartozó (`Edge`) osztály és annak függvényei.

3.1.2.4. matrix.hpp

Ebben a fájlban található a mátrixhoz tartozó osztályok és annak függvényei.

A `Matrix` osztály egyik kulcsfontosságú eleme a kettős indexelés kényelmes működése. Ehhez a mátrixot két osztályra bontottam. Az első a `Matrix` osztály. Ez `MatrixRow<T>` elemeket tárol egy listában. A második a `MatrixRow` osztály. Ez `T` -ket tárol egy listában. (Itt a T generikus adatot jelent) Ezzel a megközelítéssel a józan ész elvárásai szerint működik a kettős indexelés.

3.1.2.5. ember.h

Ez a fájl az `Ember` osztályhoz tartozó header fájl.

3.1.2.6. ember.cpp

Ez a fájl az `Ember` osztályhoz tartozó cpp fájl. Az ember osztály létezésének az oka részletesen a `Tesztesetek Dokumentációja`-ban található.

3.1.3. Memóriaszivárgás ellenőrzése

A program fejlesztése során felhasználtam a `memtrace` nevű programot, amelyet már korábban a laborok során használtam. Így végig ellenőrizni tudtam, hogy a programban nem található memóriaszivárgás.

3.1.4. A program használata

A program használatára a `Tesztesetek Dokumentációja` -ban található részletes bemutató. Ezen felül a belső működés részletes leírása a dokumentáció végén található.

4. fejezet

Tesztesetek Dokumentációja

4.1. Tesztesetek Dokumentációja

4.1.1. Az Ember osztály

A tesztekhez létrehoztam egy `Ember` osztályt, ami tetszőlegesen kicserélhető bármilyen másik osztályra is. Az `Ember` osztály két változót tartalmaz. Egy `nev` és egy `kor`. Ezekben az emberek nevét és életkorát tárolom. Ezenkívül tartalmaz az osztály még néhány alapvető függvényt is. Az osztály részletesebb leírása megtalálható a Dokumentáció második felében.

4.1.2. A main-ben található tesztesetek

A main függvényben összesen 4 db fő tesztet készítettem el. Ebből az utolsó két teszteset két darab alrészből áll.

4.1.2.1. Teszt 1

Az első teszt a fájlból történő beolvasás és az oda történő kiírás ellenőrzését szolgálja. Miután lefutott létrejön `g1save.txt` nevű fájl, aminek a tartalma megegyezik a `g1f.txt` tartalmával. Ezzel ellenőriztük, hogy a fájlba írás és onnan olvasás megfelelően működik. A konzolon megjelennek a gráfhoz tartozó adatok is ezzel ez a rész is ellenőrzésre került.

```
{c++}
    std::ifstream t1f("g1f.txt");
    std::ifstream t1fdata("g1data.txt");
    Graph<Ember> t1(t1f);
    t1.readDataFromFile(t1fdata,Ember::CreateEmber);
    t1.saveAdjMatrixToFile("g1save.txt");
    std::cout << t1;
```

4.1.2.2. Teszt 2

A második teszt arra szolgál, hogy ellenőrizzük, hogy az elkészült osztály generikus-e. Itt az `Ember` után `char` típussal hozom létre a gráfot. Itt a konzolon megjelennek a `char` típusú adatok ebből tudjuk, hogy ez is jól működik.

```
std::ifstream t2f("g2f.txt");
std::ifstream t2fdata("g2fdata.txt");
Graph<char> t2(t2f);
t2.readDataFromFile(t2fdata);
std::cout << t2;
```

4.1.2.3. Teszt 3

A harmadik tesztben a gráfon lefuttatok egy BFS algoritmust. Itt a konzolon megjelennek az algoritmus által bejárt csúcsok. Ebből tudjuk, hogy ez is jól működik.

```
std::ifstream t3f("g5f.txt");
Graph<char> t3(t3f);
Graph<char>::BFSSet *ret = t3.BFS(4);
std::cout << t3;
std::cout << *ret;
delete ret
```

4.1.2.4. Teszt 4

A negyedik tesztben az összefüggőségi vizsgálatot végzem el. Az A részben a gráf összefüggő a B részben nem. A B részben található gráf nagy mértékben hasonlít az A részben láthatóhoz. A fő különbség annyi, hogy a B gráfban a 0-ás és 1-es csúcs csak egymással vannak összekötve, ezért a gráfnak legalább 2 db komponense van, tehát nem lesz összefüggő.

4.1.2.4.1. A rész

Itt a konzolon megjelenő 1-es szám jelzi számunkra, hogy a gráf összefüggő.

```
std::ifstream t4af("g6af.txt");
std::ifstream t4afdata("g6bdata.txt");
Graph<Ember> t4a(t4af);
t4a.readDataFromFile(t4afdata,Ember::CreateEmber);
std::cout << t4a;
std::cout << t4a.isConnectedGraph() << std::endl;
```

4.1.2.4.2. B rész

Itt a konzolon megjelenő 0-ás szám jelzi, hogy a gráf nem összefüggő.

```
std::ifstream t4bf("g6bf.txt");
std::ifstream t4bfdata("g6bdata.txt");
Graph<Ember> t4b(t4bf);
t4b.readDataFromFile(t4bfdata,Ember::CreateEmber);
std::cout << t4b;
std::cout << t4b.isConnectedGraph() << std::endl;
```

Összességében úgy gondolom, hogy ezekkel a tesztekkel lefedtem a programot, azok alapján a szempontok alapján, amit a specifikációban rögzítettem.

5. fejezet

Hierarchikus mutató

5.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

6. fejezet

Adatszerkezet-mutató

6.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

7. fejezet

Fájlmutató

7.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

8. fejezet

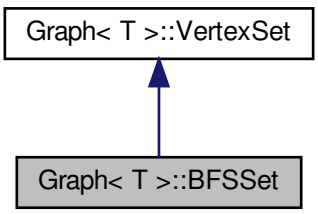
Adatszerkezetek dokumentációja

8.1. Graph< T >::BFSSet osztályreferencia

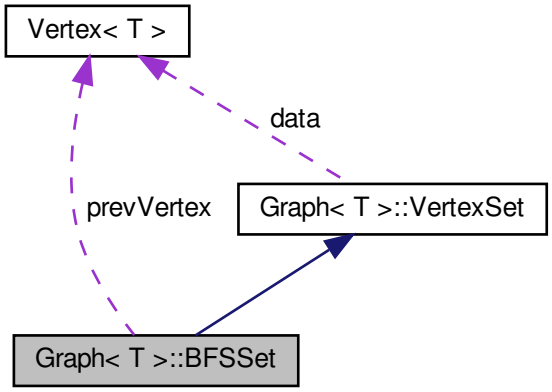
BFS algoritmushoz készült tároló. A VerexSet-ből származik.

`#include <graph.hpp>`

A Graph< T >::BFSSet osztály származási diagramja:



A Graph< T >::BFSSet osztály együttműködési diagramja:



Publikus tagfüggvények

- BFSSet ()

  BFSSet default konstruktora.
- BFSSet (size_t len)

  Max méretet beállító konstruktora.
- BFSSet (const BFSSet &bs)

  BFSSet másoló konstruktora.
- BFSSet & operator= (const BFSSet &bs)

  BFSSet értékadó operátora.
- void setDistance (size_t kinek, long mire)

  Beállítja a csúcs távolságát.
- long getDistance (size_t kinek)

  Visszaadja a csúcs távolságát a kezdőponttól mérve.
- void addPrevVertex (Vertex< T > *v)

  A PrevVertex listának a végéhez hozzáad egy elemet.
- ~BFSSet ()

  A BFSSet destruktora.

Privát attribútumok

- long * distance

  csúcs távolsága a kezdőcsúcstól
- Vertex< T > ** prevVertex

  Lista, amely minden csúcshoz hozzárendelő az őt megelőző csúcs azonosítóját.
- size_t prevVertexLen

  Tárolja a lista aktuális méretét.

Barátok

- std::ostream & operator<< (std::ostream &os, const BFSSet &bfss)

  A BFSSet << operátora.

Additional Inherited Members

8.1.1. Részletes leírás

template<typename T>
class Graph< T >::BFSSet

BFS algoritmushoz készült tároló. A VerexSet-ből származik.

8.1.2. Konstruktorok és destruktorok dokumentációja

8.1.2.1. BFSSet() [1/2]

```
template<typename T>
Graph< T >::BFSSet::BFSSet (
            size_t len )  [inline]
```

Max méretet beállító konstruktora.

Paraméterek

len	maximális méret

8.1.2.2. BFSSet() [2/2]

```
template<typename T>
Graph< T >::BFSSet::BFSSet (
            const BFSSet & bs )  [inline]
```

BFSSet másoló konstruktora.

Paraméterek

bs	a másik BFSSet

8.1.3. Tagfüggvények dokumentációja

8.1.3.1. addPrevVertex()

```
template<typename T>
void Graph< T >::BFSSet::addPrevVertex (
             Vertex< T > * v )  [inline]
```

A PrevVertex listának a végéhez hozzáad egy elemet.

Paraméterek

v	A hozzáadandó csúcsra mutató pointer

8.1.3.2. getDistance()

```
template<typename T>
long Graph< T >::BFSSet::getDistance (
             size_t kinek )  [inline]
```

Visszaadja a csúcs távolságát a kezdőponttól mérve.

Paraméterek

kinek	a megfelelő csúcs

Visszatérési érték

a távolság értéke

8.1.3.3. operator=()

```
template<typename T>
BFSSet& Graph< T >::BFSSet::operator= (
             const BFSSet & bs )  [inline]
```

BFSSet értékadó operátora.

Paraméterek

bs	a másik BFSSet

Visszatérési érték

visszaadja a keletkezett BFSSetet

8.1.3.4. setDistance()

```
template<typename T>
void Graph< T >::BFSSet::setDistance (
            size_t kinek,
            long mire )  [inline]
```

Beállítja a csúcs távolságát.

Paraméterek

kinek	a megfelelő csúcs
mire	a távolság érték

8.1.4. Barát és kapcsolódó függvények dokumentációja

8.1.4.1. operator<<

```
template<typename T>
std::ostream& operator<< (
            std::ostream & os,
            const BFSSet & bfss )  [friend]
```

A BFSSet << operátora.

Paraméterek

os	A kapott ostream
bfss	A kapott BFSSet

Visszatérési érték

A keletkezett ostream

Ez a dokumentáció az osztályról a következő fájl alapján készült:

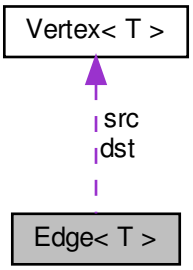
- graph.hpp

8.2. Edge< T > osztálysablon-referencia

Él osztály.

`#include <edge.hpp>`

Az Edge< T > osztály együttműködési diagramja:



Publikus tagfüggvények

- Edge ()
  Az Él default konstruktora.
- Edge (size_t id, Vertex< T > *src, Vertex< T > *dst, bool connected=false, double weight=1)
  Az Él konstruktora.
- Edge (const Edge< T > &e)
  Az Él copy konstruktora.
- Edge< T > & operator= (const Edge &e)
  Az Él értékadó operátora.
- size_t getID ()
  visszadja az él IDjét
- bool isConnected ()
  Visszadja, hogy az él be van-e húzva.
- Vertex< T > * getSource ()
  Visszadja az él forráscsúcsát.
- Vertex< T > * getDestination ()
  Visszadja az él célcsúcsát.

Privát attribútumok

- size_t id
  Az él azonosítója.
- Vertex< T > * src
  A forrás csúcs.
- Vertex< T > * dst
  A cél csúcs.
- bool connected
  Az él be van-e húzva.
- double weight
  Az él súlya.

Barátok

- template<typename F >
  std::ostream & operator<< (std::ostream &os, const Edge< F > &e)

  Az él kiírása.

8.2.1. Részletes leírás

template<typename T>
class Edge< T >

Él osztály.

Template Parameters

T	az élek típusa

8.2.2. Konstruktorok és destruktorok dokumentációja

8.2.2.1. Edge() [1/2]

```
template<typename T>
Edge< T >::Edge (
            size_t id,
            Vertex< T > * src,
            Vertex< T > * dst,
            bool connected = false,
            double weight = 1 )  [inline]
```

Az Él konstruktora.

Paraméterek

id	Az Él id-je
src	A forrás csúcsra mutató pointer
dst	A cél csúcsra mutató pointer
connected	Be van-e húzva az él?
weight	Az éj súlya

8.2.2.2. Edge() [2/2]

```
template<typename T>
Edge< T >::Edge (
            const Edge< T > & e )  [inline]
```

Az Él copy konstruktora.

Paraméterek

e	A másik él

8.2.3. Tagfüggvények dokumentációja

8.2.3.1. getDestination()

```
template<typename T>
Vertex<T>* Edge< T >::getDestination ( )  [inline]
```

Visszadja az él célcsúcsát.

Visszatérési érték

A csúcsra mutató pointer

8.2.3.2. getID()

```
template<typename T>
size_t Edge< T >::getID ( )  [inline]
```

visszadja az él IDjét

Visszatérési érték

Az ID

8.2.3.3. getSource()

```
template<typename T>
Vertex<T>* Edge< T >::getSource ( )  [inline]
```

Visszadja az él forráscsúcsát.

Visszatérési érték

A csúcsra mutató pointer

8.2.3.4. isConnected()

```
template<typename T>
bool Edge< T >::isConnected ( )  [inline]
```

Visszadja, hogy az él be van-e húzva.

Visszatérési érték

igen/nem

8.2.3.5. operator=()

```
template<typename T>
Edge<T>& Edge< T >::operator= (
            const Edge< T > & e )  [inline]
```

Az Él értékadó operátora.

Paraméterek

e	A másik él

Visszatérési érték

A keletkezett él

8.2.4. Barát és kapcsolódó függvények dokumentációja

8.2.4.1. operator<<

```
template<typename T>
template<typename F >
std::ostream& operator<< (
            std::ostream & os,
            const Edge< F > & e )  [friend]
```

Az él kiírása.

Template Parameters

F	Az él típusa

Paraméterek

os	A kapott ostream
e	A kapott él

Visszatérési érték

A keletkezett ostream

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- edge.hpp

8.3. Ember osztályreferencia

Ember osztály.

`#include <osztály>`

Publikus tagfüggvények

- Ember ()
  Az Ember osztály alapértelmezett konstruktora.
- Ember (const std::string &nev, int kor)
  Névből és korbol Embert hoz létre.
- Ember (Ember &e)
  Ember osztály másoló konstruktora.
- Ember & operator= (const Ember &e)
  Az Ember osztály értékadás operátora.

Statikus publikus tagfüggvények

- static Ember * CreateEmber (const std::string &s)
  Pontosvesszővel tagolt sorból Ember-t hoz létre.

Privát attribútumok

- std::string nev
  A név.
- int kor
  A kor.

Barátok

- std::ostream & operator<< (std::ostream &os, const Ember &e)
  Ember osztály kiíratását megvalósító függvény.

8.3.1. Részletes leírás

Ember osztály.

8.3.2. Konstruktorok és destruktorok dokumentációja

8.3.2.1. Ember() [1/2]

```
Ember::Ember (
            const std::string & nev,
            int kor )  [inline]
```

Névből és korbol Embert hoz létre.

Paraméterek

nev	A kapott név
kor	A kapott kor

8.3.2.2. Ember() [2/2]

```
Ember::Ember (
            Ember & e )  [inline]
```

Ember osztály másoló konstruktora.

Paraméterek

e	A kapott Ember

8.3.3. Tagfüggvények dokumentációja

8.3.3.1. CreateEmber()

```
Ember * Ember::CreateEmber (
            const std::string & s )  [static]
```

Pontosvesszővel tagolt sorból Ember-t hoz létre.

Paraméterek

s	A kapott sor

Visszatérési érték

A létrehozott Emberre mutató pointer

8.3.3.2. operator=()

```
Ember & Ember::operator= (
            const Ember & e )
```

Az Ember osztály értékadás operátora.

Paraméterek

e	A kapott ember

Visszatérési érték

A kapott e-vel egyenlő lesz az objektum

8.3.4. Barát és kapcsolódó függvények dokumentációja

8.3.4.1. operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Ember & e )  [friend]
```

Ember osztály kiíratását megvalósító függvény.

Paraméterek

os	ostream referencia
e	A kapott ember

Visszatérési érték

A keletkezett ostream referencia

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

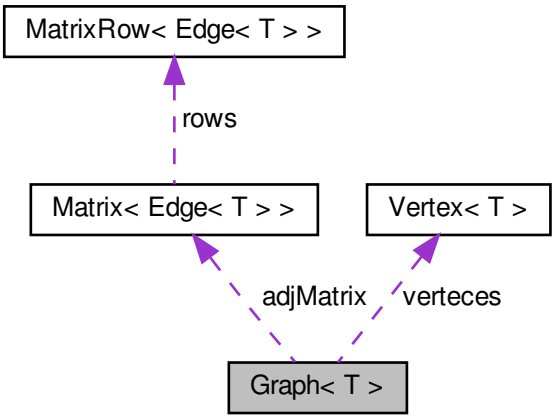
- ember.h
- ember.cpp

8.4. Graph< T > osztálysablon-referencia

Graph osztály.

`#include <graph.hpp>`

A Graph< T > osztály együttműködési diagramja:



Adatszerkezetek

- class BFSSet

  BFS algoritmushoz készült tároló. A VerexSet-ből származik.
- class VertexSet

  Csúcsok halmazának tárolására alkalmas osztály.

Publikus tagfüggvények

- Graph ()

  Grapf default konstruktora.
- Graph (std::ifstream &file)

  a Graph konstruktora fájlból
- size_t getNumberOfVertices ()

  Visszadja a csúcsok számát.
- size_t getNumberOfEdges ()

  Visszadja az élek számát.
- bool isConnectedGraph ()

Visszadja, hogy összefüggő-e a gráf, ehhez a BFS algoritmust fogja használni.

- void setEdge (size_t y, size_t x, bool con, Vertex< T > *src, Vertex< T > *dst)

  Beállít egy élet a megadott értékekre.
- Vertex< T > * getVertexFromID (size_t id)

  Visszaad egy csúcsra mutató pointert, amit ID alapján határoz meg.
- T getDataFromID (size_t id)

  Visszadja a kapott ID alapján a megfelelő csúcshoz tartozó adatot.
- BFSSet * BFS (size_t honnan=0)

  BFS alkogritmus.
- VertexSet * listNeighboursOfVertex (size_t v)

  Megkeresi a kapott csúcs szomszédait.
- void readAdjMatrixFromFile (std::ifstream &file)

  Beolvas egy szomszédsági mátrixot és beállítja azt a gráfnak.
- void saveAdjMatrixToFile (const std::string &file)

  Elmenti a szomszédsági mátrixot a megadott nevő fájlba.
- void readDataFromFile (std::ifstream &file)

  Beolvassa a csúcs adatait fájlból.
- template<class FUNC >
  void readDataFromFile (std::ifstream &file, FUNC f)

  Beolvassa a csúcs adatait fájlból, majd meghívja miden soron a kapott függvényt.
- ~Graph ()

  A gráf destruktora.

Privát attribútumok

- size_t numofv

  A csúcsok száma.
- size_t numofe

  Az élek száma.
- Matrix< Edge< T > > adjMatrix

  A gráfhoz tartozó szomszédsági mátrix.
- Vertex< T > ** verteces

  A gráfhoz tartozó csúcsok.
- bool isDataUploaded

  A gráf fel van-e töltve adatokkal.

Barátok

- template<typename F >
  std::ostream & operator<< (std::ostream &os, Graph< F > &g)

  Kirajzolja a gráf szomszédsági mátrixát.

8.4.1. Részletes leírás

template<typename T>
class Graph< T >

Graph osztály.

Template Parameters

T	a csúcsok típusa

8.4.2. Konstruktorok és destruktorok dokumentációja

8.4.2.1. Graph()

```
template<typename T>
Graph< T >::Graph (
            std::ifstream & file )  [inline]
```

a Graph konstruktora fájlból

Paraméterek

file	a szomszédsági mátrix fájlja

8.4.3. Tagfüggvények dokumentációja

8.4.3.1. BFS()

```
template<typename T>
BFSSet* Graph< T >::BFS (
            size_t honnan = 0 )  [inline]
```

BFS alkogritmus.

Paraméterek

honnan	csúcs id, ahonnan az algoritmus indul

Visszatérési érték

visszad egy BFSSet-re mutató pointert. A felszabadítás a felhasználó dolga.

8.4.3.2. getDataFromID()

```
template<typename T>
T Graph< T >::getDataFromID (
            size_t id )  [inline]
```

Visszadja a kapott ID alapján a megfelelő csúcshoz tartozó adatot.

Paraméterek

id	A kapott ID

Visszatérési érték

A csúcs adata

8.4.3.3. getNumberOfEdges()

```
template<typename T>
size_t Graph< T >::getNumberOfEdges ( )  [inline]
```

Visszadja az élek számát.

Visszatérési érték

élek száma

8.4.3.4. getNumberOfVertices()

```
template<typename T>
size_t Graph< T >::getNumberOfVertices ( )  [inline]
```

Visszadja a csúcsok számát.

Visszatérési érték

csúcsok száma

8.4.3.5. getVertexFromID()

```
template<typename T>
Vertex<T>* Graph< T >::getVertexFromID (
            size_t id )  [inline]
```

Visszaad egy csúcsra mutató pointert, amit ID alapján határoz meg.

Paraméterek

id	Az ID amit a felhasználótól kapunk

Visszatérési érték

A csúcsra mutató pointer

8.4.3.6. isConnectedGraph()

```
template<typename T>
bool Graph< T >::isConnectedGraph ( )  [inline]
```

Visszadja, hogy összefüggő-e a gráf, ehhez a BFS algoritmust fogja használni.

Visszatérési érték

összefüggő -e a gráf

8.4.3.7. listNeighboursOfVertex()

```
template<typename T>
VertexSet* Graph< T >::listNeighboursOfVertex (
            size_t v )  [inline]
```

Megkeresi a kapott csúcs szomszédait.

Paraméterek

v	a kapott csúcs

Visszatérési érték

VertexSet∗-ot ad vissza. A felszabadítás a felhasználó dolga

8.4.3.8. readAdjMatrixFromFile()

```
template<typename T>
void Graph< T >::readAdjMatrixFromFile (
            std::ifstream & file )  [inline]
```

Beolvas egy szomszédsági mátrixot és beállítja azt a gráfnak.

Paraméterek

file	A file-ra mutató ifstream

8.4.3.9. readDataFromFile() [1/2]

```
template<typename T>
void Graph< T >::readDataFromFile (
            std::ifstream & file )  [inline]
```

Beolvassa a csúcs adatait fájlból.

Paraméterek

file	A beolvasandó file

8.4.3.10. readDataFromFile() [2/2]

```
template<typename T>
template<class FUNC >
void Graph< T >::readDataFromFile (
            std::ifstream & file,
            FUNC f )  [inline]
```

Beolvassa a csúcs adatait fájlból, majd meghívja miden soron a kapott függvényt.

Template Parameters

FUNC	A kapott függvény típusa

Paraméterek

file	A kapott file
f	A kapott fügvény, amely létrehozza a T objektumot a kapott sor alapján.

8.4.3.11. saveAdjMatrixToFile()

```
template<typename T>
void Graph< T >::saveAdjMatrixToFile (
            const std::string & file )  [inline]
```

Elmenti a szomszédsági mátrixot a megadott nevő fájlba.

Paraméterek

file	A kapott fájlnév

8.4.3.12. setEdge()

```
template<typename T>
void Graph< T >::setEdge (
             size_t y,
             size_t x,
             bool con,
             Vertex< T > * src,
             Vertex< T > * dst )  [inline]
```

Beállít egy élet a megadott értékekre.

Paraméterek

y	melyik sor
x	a soron belül melyik elem
con	az él be van-e húzva
src	forráscsúcs
dst	célcsúcs

8.4.4. Barát és kapcsolódó függvények dokumentációja

8.4.4.1. operator<<

```
template<typename T>
template<typename F >
std::ostream& operator<< (
             std::ostream & os,
             Graph< F > & g )  [friend]
```

Kirajzolja a gráf szomszédsági mátrixát.

Paraméterek

os	ostream
g	a gráf

Visszatérési érték

ostream

Template Parameters

F	A gráf ilyen adatokat tárol

Paraméterek

os	ostream referencia
g	A kapott gráf

Visszatérési érték

A keletkezett ostream referencia

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- graph.hpp

8.5. Matrix< T > osztálysablon-referencia

A mátrix osztály.

`#include <matrix.hpp>`

Publikus tagfüggvények

- Matrix ()
  A mátrix default konstruktora.
- Matrix (size_t y, size_t x)
  A mátrix konstruktora.
- Matrix (std::ifstream &file)
  A mátrix konstruktora fájl alapján.
- Matrix (const Matrix< T > &m)
  A mátrix copy konstruktora.
- Matrix< T > & operator= (const Matrix< T > &m)
  A mátrix értékadó operátora.
- size_t getxmax ()
  Visszaadja a mátrix szélességét.
- size_t getymax ()
  Visszaadja a mátrix magasságát.
- MatrixRow< T > & operator[ ] (size_t i)
  Mátrix indexelő operátora.
- const MatrixRow< T > & operator[ ] (size_t i) const
  Mátrix indexelő konstans operátora.
- void setToDefaultValue (size_t y, size_t x)

Visszaállítja a mátrix celláját az alapértelmezett értékre.

- void setData (size_t y, size_t x, T mire)

  Beállítja a mátrix adatát a megadott értékre.
- template<typename FUNC >
  FUNC executeOnEveryElement (FUNC func)

  Lefutatja a kapott függvényt a mátrix összes elemén.
- void saveMatrixToFile (const std::string &filename)

  A mátrixot elmenti fájlba.
- std::ifstream & readMatrixFromFile (std::ifstream &file)

  Mátrixot beolvassa fájlból.
- ~Matrix ()

  Mátrix destruktora.

Statikus publikus tagfüggvények

- static void freeCell (T &t)

  felszabadítja a mátrix megfelelő celláját

Védett attribútumok

- size_t ymax

  A mátrix magassága.
- size_t xmax

  A mátrix szélessége.
- MatrixRow< T > * rows

  A mátrix sorait tároló lista.

Barátok

- std::ostream & operator<< (std::ostream &os, const Matrix< T > &mx)

  Mátrix kiírása.

8.5.1. Részletes leírás

template<typename T>
class Matrix< T >

A mátrix osztály.

mátrix osztály

Template Parameters

T	a mátrix adatainak típusa

8.5.2. Konstruktorok és destruktorok dokumentációja

8.5.2.1. Matrix() [1/3]

```
template<typename T>
Matrix< T >::Matrix (
            size_t y,
            size_t x )  [inline]
```

A mátrix konstruktora.

Paraméterek

y	A mátrix magassága
x	A mátrix szélessége

8.5.2.2. Matrix() [2/3]

```
template<typename T>
Matrix< T >::Matrix (
            std::ifstream & file )  [inline]
```

A mátrix konstruktora fájl alapján.

Paraméterek

file	A kapott fájl

8.5.2.3. Matrix() [3/3]

```
template<typename T>
Matrix< T >::Matrix (
            const Matrix< T > & m )  [inline]
```

A mátrix copy konstruktora.

Paraméterek

m	A kapott mátrix

8.5.3. Tagfüggvények dokumentációja

8.5.3.1. executeOnEveryElement()

```
template<typename T>
template<typename FUNC >
FUNC Matrix< T >::executeOnEveryElement (
            FUNC func )  [inline]
```

Lefutatja a kapott függvényt a mátrix összes elemén.

Template Parameters

FUNC	függvénytípus

Paraméterek

func	A kapott függvény

Visszatérési érték

A függvénytípus

8.5.3.2. freeCell()

```
template<typename T>
static void Matrix< T >::freeCell (
            T & t )  [inline], [static]
```

felszabadítja a mátrix megfelelő celláját

Paraméterek

t	A felszabadítandó adat

8.5.3.3. getxmax()

```
template<typename T>
size_t Matrix< T >::getxmax ( )  [inline]
```

Visszaadja a mátrix szélességét.

Visszatérési érték

A mátrix szélessége

8.5.3.4. getymax()

```
template<typename T>
size_t Matrix< T >::getymax ( )  [inline]
```

Visszaadja a mátrix magasságát.

Visszatérési érték

A mátrix magassága

8.5.3.5. operator=()

```
template<typename T>
Matrix<T>& Matrix< T >::operator= (
            const Matrix< T > & m )  [inline]
```

A mátrix értékadó operátora.

Paraméterek

m	A kapott mátrix

Visszatérési érték

A keletkezett mátrix

8.5.3.6. operator[]() [1/2]

```
template<typename T>
MatrixRow<T>& Matrix< T >::operator[] (
            size_t i )  [inline]
```

Mátrix indexelő operátora.

Paraméterek

i	A megfelelő sor kiválasztása

Visszatérési érték

A mátrix megfelelő sora

8.5.3.7. operator[]() [2/2]

```
template<typename T>
const MatrixRow<T>& Matrix< T >::operator[] (
            size_t i ) const  [inline]
```

Mátrix indexelő konstans operátora.

Paraméterek

i	A megfelelő sor kiválasztása

Visszatérési érték

A mátrix megfelelő sora

8.5.3.8. readMatrixFromFile()

```
template<typename T>
std::ifstream& Matrix< T >::readMatrixFromFile (
            std::ifstream & file )  [inline]
```

Mátrixot beolvassa fájlból.

Paraméterek

file	A fájlra mutató referencia

Visszatérési érték

visszadja a fájlt

8.5.3.9. saveMatrixToFile()

```
template<typename T>
void Matrix< T >::saveMatrixToFile (
            const std::string & filename )  [inline]
```

A mátrixot elmenti fájlba.

Paraméterek

filename	A kapott fájl neve

8.5.3.10. setData()

```
template<typename T>
void Matrix< T >::setData (
             size_t y,
             size_t x,
             T mire )  [inline]
```

Beállítja a mátrix adatát a megadott értékre.

Paraméterek

y	A sor kiválasztása
x	A soron belüli elem kiválasztása
mire	A beállítandó érték

8.5.3.11. setToDefaultValue()

```
template<typename T>
void Matrix< T >::setToDefaultValue (
             size_t y,
             size_t x )  [inline]
```

Visszaállítja a mátrix celláját az alapértelmezett értékre.

Paraméterek

y	A sor kiválasztása
x	A soron belüli elem kiválasztása

8.5.4. Barát és kapcsolódó függvények dokumentációja

8.5.4.1. operator<<

```
template<typename T>
std::ostream& operator<< (
```

```
std::ostream & os,
const Matrix< T > & mx )  [friend]
```

Mátrix kiírása.

Paraméterek

os	A kapott ostream
mx	A kapott mátrix

Visszatérési érték

A keletkezett ostream

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- matrix.hpp

8.6. MatrixRow< T > osztálysablon-referencia

A mátrix sora.

```
#include <matrix.hpp>
```

Publikus tagfüggvények

- MatrixRow ()
  A mátrix sorának default konstruktora.
- MatrixRow (size_t x)
  A mátrix sorának konstruktora méret alapján.
- MatrixRow (const MatrixRow< T > &mr)
  A mátrix sorának copy konstruktora.
- MatrixRow< T > & operator= (const MatrixRow< T > &mr)
  A mátrix értékadó operátora.
- T & operator[ ] (size_t x)
  A mátrix sorának indexelő operátora.
- const T & operator[ ] (size_t x) const
  A mátrix sorának indexelő konstans operátora.
- ~MatrixRow ()
  A mátrix sor destruktora.

Privát attribútumok

- T * data
  A tárolandó adatokat tartalmazó lista.
- size_t xmax
  A sor maximális mérete.

8.6.1. Részletes leírás

template<typename T>
class MatrixRow< T >

A mátrix sora.

Mátrix sorait tároló osztály.

Template Parameters

T	A mátrix sorának a típusa
T	A mátrixban tárolandó adat

8.6.2. Konstruktorok és destruktorok dokumentációja

8.6.2.1. MatrixRow() [1/2]

```
template<typename T>
MatrixRow< T >::MatrixRow (
            size_t x )  [inline]
```

A mátrix sorának konstruktora méret alapján.

Paraméterek

x	A kapott méret

8.6.2.2. MatrixRow() [2/2]

```
template<typename T>
MatrixRow< T >::MatrixRow (
            const MatrixRow< T > & mr )  [inline]
```

A mátrix sorának copy konstruktora.

Paraméterek

mr	A kapott mátrix sor

8.6.3. Tagfüggvények dokumentációja

8.6.3.1. operator=()

```
template<typename T>
MatrixRow<T>& MatrixRow< T >::operator= (
            const MatrixRow< T > & mr )  [inline]
```

A mátrix értékadó operátora.

Paraméterek

mr	A kapott mátrix sor

Visszatérési érték

A keletkezett mátrix sor

8.6.3.2. operator[]() [1/2]

```
template<typename T>
T& MatrixRow< T >::operator[] (
            size_t x )  [inline]
```

A mátrix sorának indexelő operátora.

Paraméterek

x	melyik elem a sorban

Visszatérési érték

Az adatra mutató referencia

8.6.3.3. operator[]() [2/2]

```
template<typename T>
const T& MatrixRow< T >::operator[] (
            size_t x ) const  [inline]
```

A mátrix sorának indexelő konstans operátora.

Paraméterek

x	melyik elem a sorban

Visszatérési érték

Az adatra mutató referencia

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- matrix.hpp

8.7. Vertex< V > osztálysablon-referencia

A csúcs osztály.

`#include <vertex.hpp>`

Publikus tagfüggvények

- Vertex ()
  A csúcs default konstruktora.
- Vertex (size_t id)
  A csúcs konstruktora.
- Vertex (size_t id, V &data)
  A csúcs konstruktora.
- Vertex (const Vertex &v)
  A csúcs copy konstruktora.
- Vertex & operator= (Vertex &v)
  A csúcs értékadó operátora.
- size_t getID ()
  Visszadja a csúcs azonosítóját.
- void setData (V d)
  Beállítja a csúcs adatát a kapott értékre.
- V & getData ()
  Visszadja a csúcs adatát.

Privát attribútumok

- size_t id
  A csúcs azonosítója.
- V data
  A csúcshoz tartozó adat.

8.7.1. Részletes leírás

template<typename V>
class Vertex< V >

A csúcs osztály.

Template Parameters

V	

8.7.2. Konstruktorok és destruktorok dokumentációja

8.7.2.1. Vertex() [1/3]

```
template<typename V>
Vertex< V >::Vertex (
            size_t id )  [inline]
```

A csúcs konstruktora.

Paraméterek

id	a kapott azonosító

8.7.2.2. Vertex() [2/3]

```
template<typename V>
Vertex< V >::Vertex (
            size_t id,
            V & data )  [inline]
```

A csúcs konstruktora.

Paraméterek

id	A kapott azonosító
data	A kapott adat

8.7.2.3. Vertex() [3/3]

```
template<typename V>
Vertex< V >::Vertex (
            const Vertex< V > & v )  [inline]
```

A csúcs copy konstruktora.

Paraméterek

v	A kapott csúcs

8.7.3. Tagfüggvények dokumentációja

8.7.3.1. getData()

```
template<typename V>
V& Vertex< V >::getData ( )  [inline]
```

Visszadja a csúcs adatát.

Visszatérési érték

A csúcs adata

8.7.3.2. getID()

```
template<typename V>
size_t Vertex< V >::getID ( )  [inline]
```

Visszadja a csúcs azonosítóját.

Visszatérési érték

A csúcs azonosítója

8.7.3.3. operator=()

```
template<typename V>
Vertex& Vertex< V >::operator= (
            Vertex< V > & v )  [inline]
```

A csúcs értékadó operátora.

Paraméterek

v	A kapott csúcs

Visszatérési érték

A keletkezett csúcs

8.7.3.4. setData()

```
template<typename V>
void Vertex< V >::setData (
            V d )  [inline]
```

Beállítja a csúcs adatát a kapott értékre.

Paraméterek

d	A kapott érték

Ez a dokumentáció az osztályról a következő fájl alapján készült:

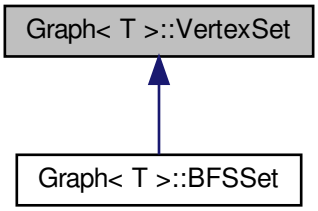
- vertex.hpp

8.8. Graph< T >::VertexSet osztályreferencia

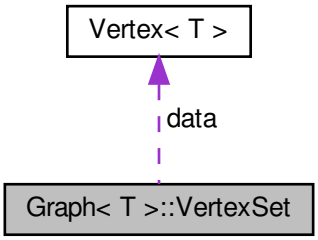
Csúcsok halmazának tárolására alkalmas osztály.

`#include <graph.hpp>`

A Graph< T >::VertexSet osztály származási diagramja:



A Graph< T >::VertexSet osztály együttműködési diagramja:



Publikus tagfüggvények

- VertexSet ()

  A csúcshalmaz default konstruktora.
- VertexSet (size_t len)

  A csúcshalmaz méret alapján történő konstruktora. Itt a méret a halmaz méretét jelenti.
- VertexSet (const VertexSet &vs)

  A halmaz copy ctorja.
- VertexSet & operator= (const VertexSet &vs)

  A halmaz = operatora.
- Vertex< T > * getVertex (size_t index)

  visszaadja a kért csúcsot
- size_t getLen ()

  visszaadja a set aktuális méretét
- void add (Vertex< T > *v)

  Hozzáad a halmazhoz egy új csúcsot.
- virtual ~VertexSet ()

  A halmaz osztály destruktora. A csúcsok felszabadítását a Gráf végzi.

Védett attribútumok

- size_t len

  Vertex set aktuális mérete.
- Vertex< T > ** data

  Az adatokat tároló tömb.

8.8.1. Részletes leírás

template<typename T>
class Graph< T >::VertexSet

Csúcsok halmazának tárolására alkalmas osztály.

8.8.2. Konstruktorok és destruktorok dokumentációja

8.8.2.1. VertexSet() [1/2]

```
template<typename T>
Graph< T >::VertexSet::VertexSet (
            size_t len )  [inline]
```

A csúcshalmaz méret alapján történő konstruktora. Itt a méret a halmaz méretét jelenti.

Paraméterek

len	A halmaz mérete

8.8.2.2. VertexSet() [2/2]

```
template<typename T>
Graph< T >::VertexSet::VertexSet (
            const VertexSet & vs )  [inline]
```

A halmaz copy ctorja.

Paraméterek

vs	A halmaz, amit másolni szeretnénk.

8.8.3. Tagfüggvények dokumentációja

8.8.3.1. add()

```
template<typename T>
void Graph< T >::VertexSet::add (
            Vertex< T > * v )  [inline]
```

Hozzáad a halmazhoz egy új csúcsot.

Paraméterek

v	a csúcsra mutató pointer

8.8.3.2. getLen()

```
template<typename T>
size_t Graph< T >::VertexSet::getLen ( )  [inline]
```

visszaadja a set aktuális méretét

Visszatérési érték

a méret

8.8.3.3. getVertex()

```
template<typename T>
Vertex<T>* Graph< T >::VertexSet::getVertex (
            size_t index )  [inline]
```

visszaadja a kért csúcsot

Paraméterek

index	a csúcs id-je

Visszatérési érték

a kért csúcs

8.8.3.4. operator=()

```
template<typename T>
VertexSet& Graph< T >::VertexSet::operator= (
            const VertexSet & vs )  [inline]
```

A halmaz = operatora.

Paraméterek

vs	az egyenlőségjel jobb oldalán lévő halmazra mutató referencia

Visszatérési érték

Az új halmaz

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- graph.hpp

9. fejezet

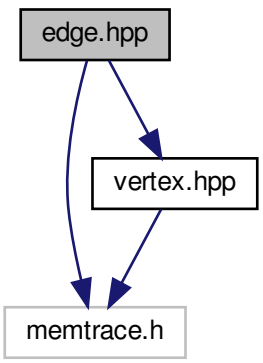
Fájlok dokumentációja

9.1. edge.hpp fájlreferencia

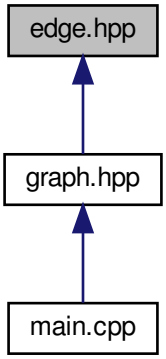
edge header

```
#include "memtrace.h"
#include "vertex.hpp"
```

Az edge.hpp definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- class Edge< T >
  Él osztály.

Függvények

- template<typename F >
  std::ostream & operator<< (std::ostream &os, const Edge< F > &e)
  Az él kiírása.

9.1.1. Részletes leírás

edge header

9.1.2. Függvények dokumentációja

9.1.2.1. operator<<()

```
template<typename F >
std::ostream& operator<< (
            std::ostream & os,
            const Edge< F > & e )
```

Az él kiírása.

Template Parameters

F	Az él típusa

Paraméterek

os	A kapott ostream
e	A kapott él

Visszatérési érték

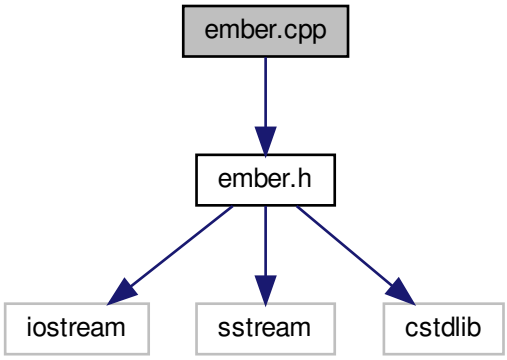
A keletkezett ostream

9.2. ember.cpp fájlreferencia

ember.cpp kód

```
#include "ember.h"
```

Az ember.cpp definíciós fájl függési gráfja:



Függvények

- std::ostream & operator<< (std::ostream &os, const Ember &e)

  Ember osztály kiíratását megvalósító függvény.

9.2.1. Részletes leírás

ember.cpp kód

9.2.2. Függvények dokumentációja

9.2.2.1. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Ember & e )
```

Ember osztály kiíratását megvalósító függvény.

Paraméterek

os	ostream referencia
e	A kapott ember

Visszatérési érték

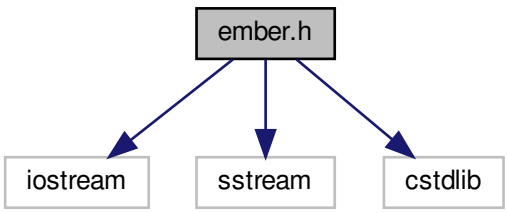
A keletkezett ostream referencia

9.3. ember.h fájlreferencia

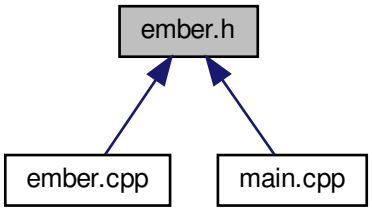
ember.h header

```
#include <iostream>
#include <sstream>
#include <cstdlib>
```

Az ember.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- class Ember

  Ember osztály.

9.3.1. Részletes leírás

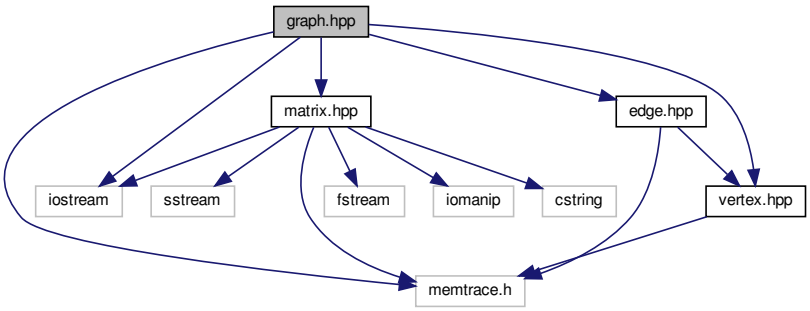
ember.h header

9.4. graph.hpp fájlreferencia

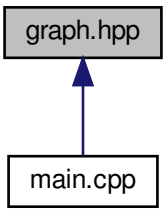
graph.h header

```
#include <iostream>
#include "memtrace.h"
#include "matrix.hpp"
#include "edge.hpp"
#include "vertex.hpp"
```

A graph.hpp definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- class Graph< T >
  Graph osztály.
- class Graph< T >::VertexSet
  Csúcsok halmazának tárolására alkalmas osztály.
- class Graph< T >::BFSSet
  BFS algoritmushoz készült tároló. A VerexSet-ből származik.

Függvények

- template<typename F >
  std::ostream & operator<< (std::ostream &os, Graph< F > &g)
  A gráf kiírását végző függvény.

9.4.1. Részletes leírás

graph.h header

9.4.2. Függvények dokumentációja

9.4.2.1. operator<<()

```
template<typename F >
std::ostream& operator<< (
            std::ostream & os,
            Graph< F > & g )
```

A gráf kiírását végző függvény.

Kirajzolja a gráf szomszédsági mátrixát.

Template Parameters

F	A gráf ilyen adatokat tárol

Paraméterek

os	ostream referencia
g	A kapott gráf

Visszatérési érték

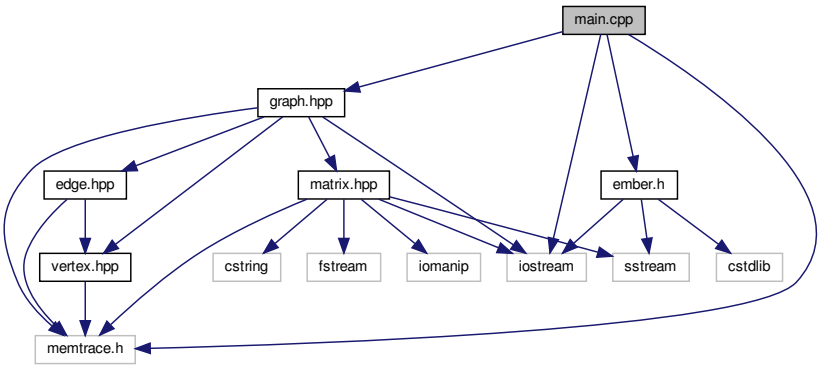
A keletkezett ostream referencia

9.5. main.cpp fájlreferencia

main.cpp forrás

```
#include <iostream>
#include "memtrace.h"
#include "graph.hpp"
#include "ember.h"
```

A main.cpp definíciós fájl függési gráfja:



Függvények

- int **main** ()

9.5.1. Részletes leírás

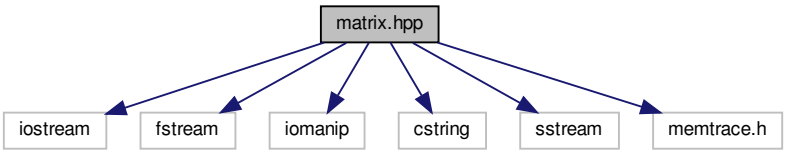
main.cpp forrás

9.6. matrix.hpp fájlreferencia

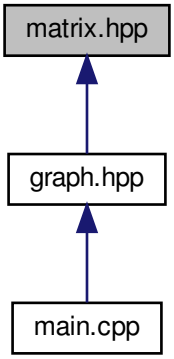
matrix header

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cstring>
#include <sstream>
#include "memtrace.h"
```

A matrix.hpp definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- class Matrix< T >
  A mátrix osztály.
- class MatrixRow< T >
  A mátrix sora.
- class Matrix< T >
  A mátrix osztály.
- class MatrixRow< T >
  A mátrix sora.

9.6.1. Részletes leírás

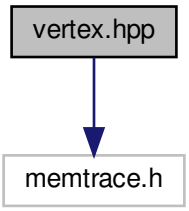
matrix header

9.7. vertex.hpp fájlreferencia

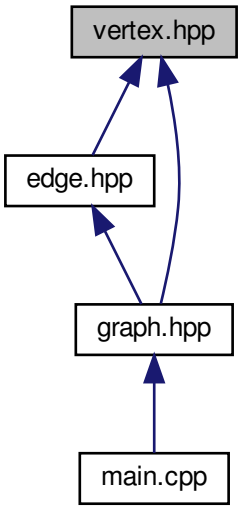
vertex header

```
#include "memtrace.h"
```

A vertex.hpp definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Adatszerkezetek

- class Vertex< V >

  A csúcs osztály.

9.7.1. Részletes leírás

vertex header

Tárgymutató