Generikus irányítatlan (egyszerű) Gráf

- Specifikáció -

# Feladatleírás:

Készítsen generikus irányítatlan (egyszerű) 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!

# ***Pontosítás***:

## Tárolás:

A feladatleírásnak megfelelően a gráf megadása szomszédsági mátrixszal történik, viszont a tárolása már nem. Minden csúcsból el lehet majd érni annak szomszédjait, illetve a Gráf osztályból minden csúcsot szintén el lehet érni.

## Input:

Egy gráf létrehozásához szükséges:  Csúcsok száma: N.

* Adott csúcsnak az értéke: N elemű egydimenziós tömb.
* A szomszédsági mátrix: N\*N es kétdimenziós tömb.

## Funkciók:

Ahogy a feladat is írja, képes lesz a csúcsainak, éleinek számát vissza adni. Emellett képes saját magáról eldönteni, hogy összefüggő-e. Emellett a gráfot lehet majd indexelni, amely az input második sorának megfelelően fog működni. Ekkor a megfelelő adat tagot adja vissza. Egy csúcsnak ki lehet majd íratni a szomszédjait illetve azok értékét. Egy gráfot is ki lehet majd íratni, mely azt jelenti, hogy soronként egy csúcsnak a szomszédjait írja, ki úgy mintha csak egy csúcsot íratnánk ki. Egy gráfnak le lehet majd generálni a szomszédsági mátrixát is.

Generikus irányítatlan (egyszerű) Gráf

- Terv -

# UML osztály diagram:

# Osztályok és adattagjai- / függvényeinek részletes leírása:

## Osszefugg enumeráció:

### Leírás:

3 értéke lehet, nagyon hasonló a boolhoz.

### Lehetséges értékek:

* igen: A bool típus true értékéhez hasonló.
* nem: A bool típus false értékéhez hasonló.
* nincsinf: Azt jelenti, hogy valamiről nincs információ (se nem true, se nem false).

## BFS\_lepes struktúra:

### Leírás:

Egy BFS bejárás lépéséről tud tárolni információt.

### Adattagok:

* csucs: Csucs típusú pointer, mely az adott lépésben érintett csúcs pointerét tárolja.
* tavolsag: A BFS algoritmus kezdő csúcsától lévő távolság.

## Graf osztály:

### Leírás:

A gráf főosztálya. Ebben tárolódik a gráfunk és a gráfon végezhető művetek.

### Template:

* T: typename típusú, a gráf generikuságát teszi lehetővé. A csúcsok által tárolt adatnak a típusát lehet vele megadni.
* N: size\_t típusú, a csúcsok számát adja meg, ezáltal a gráfot definiáló mátrix mérete is, illetve az adatokat megadó tömb mérete is.

### Adattagok:

* csucsok: Egy Csucs\* típusú egydimenziós tömb, melyben a gráf csúcsainak pointer-e tárólódik, a megadott mátrix/adat tömb sor indexeinek sorrendjében.
* csucsDb: Egy size\_t típusú változó mely a csúcsok számát tárolja (a felhasználó által megadott N-nel egyenlő).
* elekDb: Egy size\_t típusú vátozó, amely az élek számát tárolja, ha még nem kérték, hogy számolja ki (getEldb()-el nem hivatkoztak rá), akkor az értéke a maximális lehetséges élszámnál nagyobb () értéket vesz fel.
* osszeFugges: Ennek a típusa az osszefugg nevű enumeráció, mely a gráf összefüggéséről tárol „állítást”. Az „igen-nem” megfeleltethető a logikai „igaz-hamis” -nak. Értéke „nincsinf”, ha a felhasználó még nem „kérdezte” meg hogy osszefüggő-e a gráf (osszefuggo() fv. -t nem híta még meg a gráfra).
* matrix: bool típusú kétdimenziós tömb, amely a gráf szomszédsági mátrixát tárolja, null ha még nem kérték le a (getMatrix) fv.-el.

### Függvények:

* Graf(const T\* adatok, const bool\*\* matrix): Konstruktor, argumentumai:
  + adatok: T típusú egydimenziós tömb, mely a megfelelő indexű csúcsoknak az adat tagját tartalmazza.
  + matrix: bool típusú kétdimenziós tömb, mely a gráfot definiáló szomszédságimátrixot tartalmazza. A sorai és oszlopai indexe a megfelelő indexű csúcsnak felelnek meg és matrix[i][j] (i: sor index, j: oszlopindex) true, ha i. csúcs és j. csúcs között fut él, tehát szomszédosak.
  + Graf(const Graf& other): Másoló konstruktor az Graf típusú other objektum másolatát hozza létre.
* void bejar(size\_t csucsi, BFS\_lepes\* ut): A gráf „csucsi” indexű csúcsából indulva lefuttat egy BFS algoritmust (szélességi bejárást), kivételt dob ha az index nem létezik, argumentumai:
  + csucsi: Kezdő csúcs indexe.
  + Út: BFS\_lepes típusú egydimenziós tömb, melyet a felhasználónak kell létrehoznia, a „bejar” fv., ebben fogja a lépéseket tárolni.
* bool osszefuggo(): A gráf összefüggését jellemző logikai értéket adja vissza. Első megívásra ez a fv. számítja ki, hogy összefüggő-e és a osszeFugges adattagban eltárolja azt.
* size\_t getCsucsdb(): A csucsDb adattag gettere.
* size\_t getElekdb(): Az elekDb gettere(), ha a elekDb nagyobb a maximálisan lehetséges élszámnál elsőnek kiszámolja ezt az értéket és eltárolja az elekDb-ban.
* bool\*\* getMatrix(): A matrix adattag gettere, ha null akkor elsőnek létre hozza a szomszédsági mátrixot és eltárolja a mutatóját a matrix adattagban.
* Csucs\* operator [](size\_i i): Indexelő operátor, a gráf megfelelő indexű csúcsát adja vissza, kivételt dob érvénytelen index esetén.
* ~Graf(): Destruktor, felszabadítja a dinamikusan lefoglalt memória területeket:
  + csucsok
  + matrix

## Csucs osztály leírása:

### Leírás:

A csúcsok adatai tárolására szolgáló osztály.

### Template:

* T: typename típusú, a csúcsot generikuságát teszi lehetővé. Az általuk tárolt adatnak a típusát lehet vele megadni.
* MaxFok: a maximális fokszámot lehet megadni.

### Adattagok:

* adat: T típusú változó, amely az adott csúcs által tárolt adatot tárolja.
* szomszedok: Csucs\* típusú egydimenziós tömb. Az adott csúcs szomszédainak pointereit tárolja.
* fokaszam: size\_t típusú, az adot csúcs fokszámát tárolja.

### Fügvényei:

* Csucs(T adat = 0): Konstruktor, ami egy megadott T típusú adattal létrehoz egy csúcsot. Alapértelmezett értéke 0, így lehet tömböt létrehozni belőle.
* Csucs(Csucs& const other): Másoló konstruktor az Csucs típusú other másolatát hozza létre.
* T getAdat(): Az adat adattag getter.
* setAdat(T adat): Az adat adattag értékének a függvény argumentumaként megadott adat értékét adja.
* bool (const Csucs\* cs): Az adott csúcs szomszédjaihoz adja a megadott „cs” csúcsot, a visszatérési értéke jelzi a művelet sikerességét.
* size\_t findSzomszed(const Csucs\* cs): Az adott csúcs szomszédai között keresi a megadott „cs” csúcsot, ha megtalálta a „cs” szomszedok adattagbeli indexét adja vissza, ha nincs benne akkor a maxFokszamot.
* bool remove(const Csucs\* cs): Az adott csúcs szomszedok adattagjából eltávolítja az adott „cs” csúcsot, a visszatérési értéke jelzi, hogy sikeres volt e a művelet.
* size\_t getFok(): a fokszam getttere.
* Csucs\* operator [](size\_t i): Indexelő operátor a szomszedok adattag i. tagját adja vissza. Kivételt dob ha érvénytelen az index.
* ~Csucs(): Destruktor, felszabadítja a dinamikusan foglalt területeket:
  + szomszedok