

# A PROGRAMOZÁS ALAPJAI 2.

HÁZI FELADAT DOKUMENTÁCIÓ

## ÚTVONALKERESÉS

KÉSZÍTETTE: HIRTH BALÁZS, QIHLOP  
[hirth.balazs@gmail.com](mailto:hirth.balazs@gmail.com)

KÉSZÍTÉS FÉLÉVE: 2017/18/2

# TARTALOMJEGYZÉK

Felhasználói dokumentáció .....	3
Osztályok statikus leírása .....	3
Jarat .....	3
Felelőssége .....	3
Attribútumok .....	3
Metódusok .....	3
Variable .....	4
Felelőssége .....	4
Ősosztályok .....	4
Attribútumok .....	4
Metódusok .....	4
Út .....	4
Felelőssége .....	4
Attribútumok .....	4
Metódusok .....	4
UML osztálydiagramm .....	5
Összegzés .....	5
Mit sikerült és mit nem sikerült megvalósítani a specifikációból? .....	5
Mit tanultál a megvalósítás során? .....	5
Továbbfejlesztési lehetőségek .....	5
Képernyőképek a futó alkalmazásról .....	6

## Felhasználói dokumentáció

A programnak szöveges fájlban meg kell adni a lehetséges útvonalszakaszokat. Fontos, hogy az úthálózat gráfja összefüggő legyen, a program nem tudja értelmezni az elszigetelt területeket. A szöveges fájl tartalma soronként egy kiindulási és egy célállomás, melyeket egész számértékként adhatunk meg szóközzel elválasztva (feltételezzük, hogy a járatok mindkét irányba közlekednek, ezért ha megadunk egy 1-ből 2-be közlekedő járatot, automatikusan létezni fog egy 2-ből 1-be közlekedő is). A program tehát a gráf éleit tárolja, nem pedig az állomásokat.

A program indítása után meg kell adni egy kezdő és egy végpontot, ezután a megfelelő műveletek végrehajtódnak és visszakapjuk az egyik lehetséges legkevesebb átszállást tartalmazó útvonalat, illetve az útvonalon közlekedő járművek paramétereit. A járatok megnevezése a kezdő-és végállomásuk kiírásával történik pl.: 1 2

A megadott teszhálózatra egy lehetséges bemenet: 4 és 6. Ami azt jelenti, hogy 4-ből szeretnénk 6-ba menni a legrövidebb úton (legkevesebb átszállással). Természetesen létezhet több egyformán jó út is, ilyenkor az információk sorrendjétől függően egy lehetséges variációt kapunk vissza. A kimenet ebben az esetben az 5-6, 1-5, 1-4 érintett élek információi, illetve a gráf csúcspontjának (vagyis a városok) sorrendje (4-1-5-6).

A program két fő függvényből áll, melyeket a létrehozott egyetlen Út objektumon hívunk meg automatikusan a beírt két paraméter után (a két paraméterrel hozzuk létre az objektumot). Az első függvény dolga, hogy beolvassa a fájlból az adatokat és miközben létrehozza a Járat objektumokat, egy dinamikus növekvő tömbben tárolja el azokat. A második függvény a szélességi keresést végzi ezen a tömbön. A szélességi keresésnél folyamatosan nyilvántartjuk a már vizsgált objektumokat, hogy egy Járat objektum csak egyszer legyen felhasználható és ezeket a kiindulópontához képesti mélységük szerint sorrendbe rakva beletesszük egy új tömbbe, ahonnan már egyértelműen meghatározható az útvonal.

## Osztályok statikus leírása

### Jarat

#### Felelőssége

Egy járatot (gráf élt) reprezentáló osztály.

#### Attribútumok

*Protected*

**checked:** arra szolgál, hogy egy járatot a keresés során csak egyszer vizsgáljunk

**from:** a járat innen közlekedik

**to:** a járat eddig közlekedik

#### Metódusok

Az **virtual int getFrom()** és **virtual int getTo()** paraméter nélküli függvények lekérdezik a keresett kezdő és végpontot.

A **virtual void printInfo()** függvény kiírja a megfelelő információkat a járatról.

A **checkThis()** megjelöli a már vizsgált járatot, a **getChecked()**-del pedig lekérdezhetjük az aktuális státuszát.

## Variable

### Felelőssége

Egy járatot (gráf élt) reprezentáló osztály. Ennek segítségével képesek vagyunk a járatokkal kapcsolatos extra szolgáltatásokat és információkat közölni a felhasználóval.

### Össztályok

**Járat:** Mivel a Variable osztály a Járat viselkedését bővíti ki, ezért azt a döntést hoztam meg, hogy a Variable a Járatból öröklődjön.

### Attribútumok

#### Privát

- **from:** a járat innen közlekedik
- **to:** a járat eddig közlekedik
- **checked:** arra szolgál, hogy egy járatot csak egyszer vizsgáljunk a keresés során
- **WiFi:** extra szolgáltatást jelző változó (értéke 0/1 (nincs/van))
- **Class:** extra szolgáltatást jelző változó (értéke 0/1 (nincs/van))

} Protectedként  
öröklődnek az  
össztályból

### Metódusok

#### Publikus

Az **int getFrom()** és **int getTo()** paraméter nélküli függvények lekérdezik a keresett kezdő és végpontot.

A **void printInfo()** függvény kiírja a megfelelő információkat a járatról.

A **checkThis()** megjelöli a már vizsgált járatot, a **getChecked()**-del pedig lekérdezhethetjük az aktuális státuszát.

## Út

### Felelőssége

Ez az osztály képes eltárolni a létrejövő járatokat és szélességi keresést végrehajtani rajtuk, végül az eredményt ki is írja.

### Attribútumok

#### Privát

**honnan:** a szélességi keresést ettől a ponttól kezdjük

**hova:** a legrövidebb utak fájában pedig ezt a pontot keressük

**count:** a dinamikus tömb méretét tárolja

### Metódusok

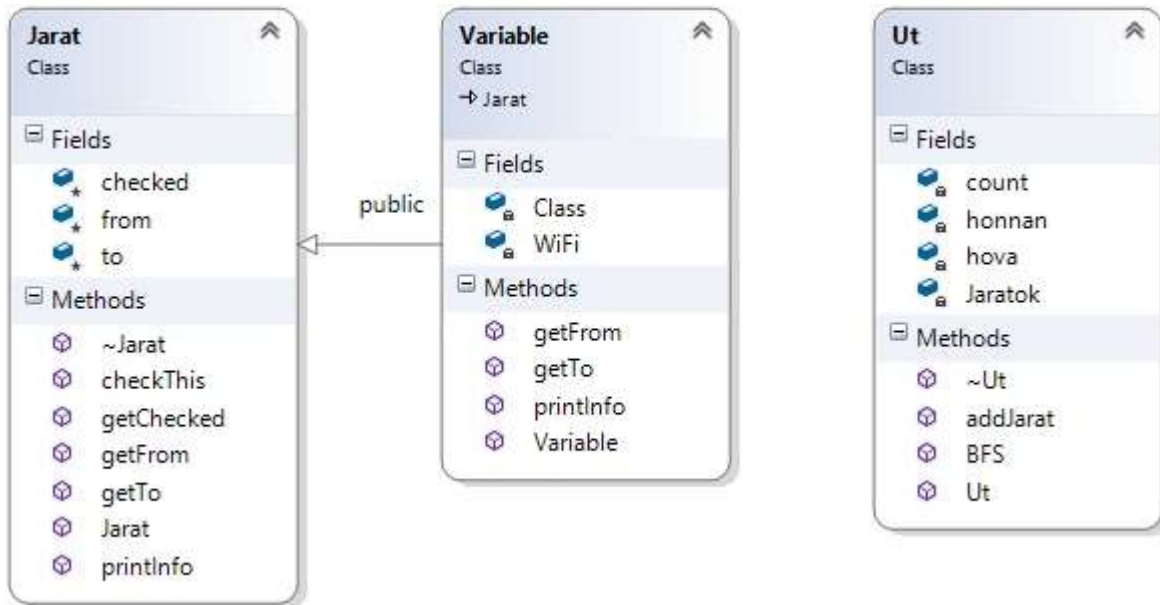
#### Publikus

**Ut(int honnan, int hova)** a konstruktorának két paramétere a kezdő és végpont

**void addJarat()** felelős a dinamikus tömb fájlból való feltöltéséért

**void BFS()** végzi a keresést, illetve az eredményt ki is írja.

# UML osztálydiagramm



## Összegzés

### Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Specifikációhoz képest a költségfüggvényeket nem tudtam megvalósítani idő hiányában. A szélességi kereséshez később jöttem rá a feladathoz illő megfelelő metodikára.

Ezt leszámítva mindent sikerült megvalósítani.

### Mit tanultál a megvalósítás során?

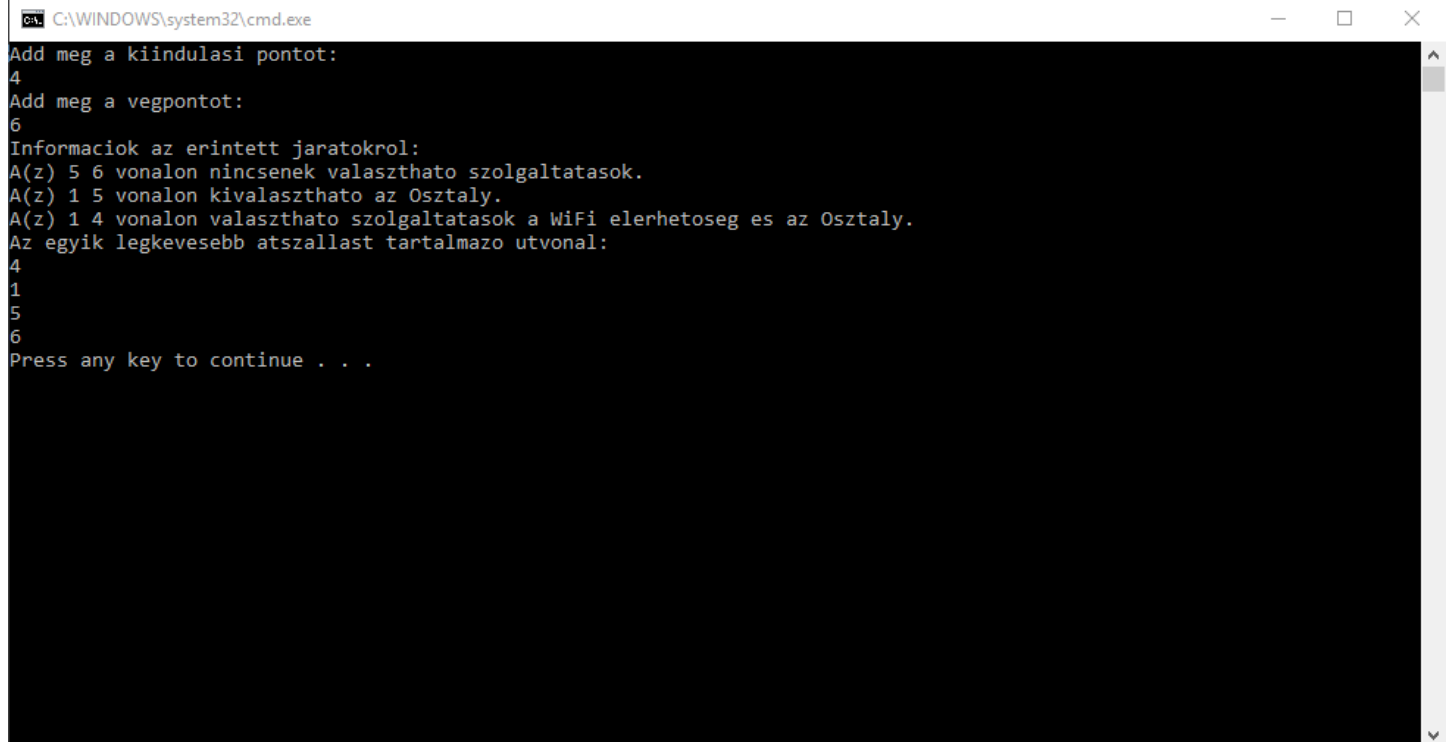
Az egységbezárás hasznossága az egyik legfontosabb amit tanultam. A gyakran átláthatatlan kódot egyértelműbbé és könnyebben használhatóbbá tette a legtöbb esetben. Nehézséget főleg a pointerműveletek okoztak, illetve a hibás pointerműveletekből adódó memóriakezelési hibák.

### Továbbfejlesztési lehetőségek

Az alkalmazás továbbfejlesztésére egy jó lehetőség, ha kibővítjük az algoritmusokat élsúlyozott gráfokra is, így valószínűbb útvonalakat is lehetne keresni, hiszen a legkevesebb átszállás nem mindenképpen a legrövidebb utat is jelenti.

Célközönség bővítése lehet a csak autóval közlekedők, illetve a csak tömegközlekedők különválasztása. Pl aki tömegközlekedik, annak ne ajánljon autótutat, de aki autóval menne, annak ne ajánljon villamosvonalat (csak ha azon az úton közlekedve kedvezőbb a helyzete).

## Képernyőképek a futó alkalmazásról



```
C:\WINDOWS\system32\cmd.exe
Add meg a kiindulasi pontot:
4
Add meg a vegpontot:
6
Informaciok az érintett jaratokrol:
A(z) 5 6 vonalon nincsenek valaszthato szolgaltatasok.
A(z) 1 5 vonalon kivalaszthato az Osztaly.
A(z) 1 4 vonalon valaszthato szolgaltatasok a WiFi elerhetoseg es az Osztaly.
Az egyik legkevesebb atszallast tartalmazo utvonal:
4
1
5
6
Press any key to continue . . .
```