Tartalom

[Tartalom 1](#_Toc456456829)

[Cél 2](#_Toc456456830)

[A program 3](#_Toc456456831)

[Követelmények 3](#_Toc456456832)

[Felhasznált eszközök 3](#_Toc456456833)

[A program szerkezete 4](#_Toc456456834)

[JavaScript fájlok 5](#_Toc456456835)

[Fejlesztési lehetőségek 6](#_Toc456456836)

[Tanulási terv 7](#_Toc456456837)

[Új anyag készítése 7](#_Toc456456838)

[Példa – n faktoriális 8](#_Toc456456839)

# Cél

Az volt a célom ennek az alkalmazásnak az elkészítésével, hogy az egyes programozási tételeket, rendezéseket, adatszerkezeteket vizuális megjelenítéssel, illetve az algoritmusok lépésenkénti megtekintésével könnyebben meg lehessen érteni, tanulni, át lehessen látni.

Emellett úgy fejlesztettem le az egyes funkciókat, hogy a hozzájuk tartozó függvényeket, eljárásokat könnyen újra lehessen használni, ezáltal könnyen tehetünk a weboldalra újabb algoritmusokat, amiket szeretnénk bemutatni, megtanítani a programom felhasználásával.

Az ELTE Informatika Karán szereplő Programozási alapismeretek című kurzushoz is segédanyagként felhasználható lehet, hiszen részben fedi a tárgyhoz szükséges alap tudásanyagot. A kurzusról annyit kell tudni, hogy a programozás alapjainak megtanítása a célja egy általános programozási nyelven keresztül (pszeudokód/struktogram), majd az algoritmusok kipróbálása, implementálása C++-ban történik. Így a kurzus végén a hallgatók elsajátítják a programozáshoz szükséges algoritmikus gondolkodást, illetve egy kis tárgyi tudást is szereznek a mai programozási nyelvek struktúrájából.

# A program

## Követelmények

Nincs túl nagy követelménye a programnak, elég hozzá egy böngésző (Internet Explorer, Mozilla FireFox, Google Chrome) és akadály nélkül használható. Teljesen böngésző, platform, operációs rendszer függetlenül csináltam meg.

A jQuery követelményeit, támogatottságát alapul véve (<https://jquery.com/browser-support/)> a következő böngészőkön biztosan működik: Internet Explorer 9+, Google Chrome 50.x-51.x, Mozilla FireFox 46.x-47.x

## Felhasznált eszközök

Úgy csináltam meg az alkalmazást, hogy ne kelljen hozzá külön szervert létesíteni, így akár lokálisan, ha egy diák/tanár letölti a saját gépére akkor is ki tudja próbálni. Az oldal létrehozásához mindössze HTML, CSS, illetve az oldalon történő eseményekhez JavaScript programozási nyelveket használtam.

A weblap készítése során több külsős, ingyenesen felhasználható keretrendszert is felhasználtam. Ezekről részletesen nem írók, de pár gondolatot mindenképpen érdemes róluk mondani.

**jQuery** (2.1.4) JavaScript keretrendszerrel a DOM-ban (Document Object Model) lévő elemeket könnyebben, gyorsabban el lehet érni, illetve azok tulajdonságait átállítani. Rengeteg hasznos függvény van benne, ami a felhasználói élményt növeli, de mindezt kevés és érthető kódolással.  
(Honlap: <https://jquery.com/>)

Az **i18next** JavaScript keretrendszer, ami a nyelvi elemek kezelését segíti, ezáltal könnyen lehet az oldalt nyelvi fájlokkal lefordítani. Az alapoldal megszerkesztése bonyolultabbá válik, azonban ezután a különböző nyelvekre való fordítás jóval egyszerűbb, könnyebb lesz. Így később a fordításhoz nem is kell programozói ismeret elég egy fájlban lévő szöveges elemeket lefordítani.  
(Honlap: <http://i18next.com/>)

**Bootstrap** rengeteg CSS tulajdonságot, class-okat tartalmaz, ami az egységes, mai szemmel szépnek mondható kinézetet adja. Illetve glyphiconokat tartalmaz, amik egy-egy oldalon lévő elemeknek még több jelentést tud adni, jobban szimbolizálja a gomb a jelentését.  
(Honlap: <http://getbootstrap.com/>)

Emellett létrehoztam a programnak egy **git** repo-t (<https://github.com/imrebende/algotan>) is a github-on. Ezt azért is tartom jó ötletnek, mert így ténylegesen teljesen nyílt forráskódú az alkalmazás, bárki elérheti. De nem csak megnézhető, hanem akár bug-okat is lehet bejelenteni rajta, így mindig naprakész lesz. Plusz a később bemutatott egyedi algoritmus fejlesztésével akár a diákok saját kódjukat is feltölthetik, így egy egyre növekvő kódgyűjtemény lesz és még a diákot is büszkeséggel töltheti el, hogy mások is megnézhetik az ő kreálmányát.

Használtam az UglifyJS 2 nevű JavaScript minifier-t, amivel könnyen csökkenteni tudtam a JS fájlok méretét, ezzel csökkentve az oldalt betöltődési idején. Illetve így az alkalmazás kisebb helyen is elfér.  
(Honlap: <https://github.com/mishoo/UglifyJS2>)

## A program szerkezete

A gyökérben helyezkednek el a HTML fájlok, minden egyes algoritmusnak, adatszerkezetnek külön fájljai, oldalai vannak. Itt van még a „*.htaccess”* fájl is, amivel cache-elési tulajdonságokat, illetve gzip tömörítést állítottam be a szerver kisebb leterheltsége érdekében.

*cpp* könyvtár tartalmaz C++ minta programokat, minden egyes algoritmushoz. Melyek megfelelő fordítóval egyből kipróbálhatóak.

*css* mappában szerepelnek a CSS fájlok (ebből mindössze kettő van a Bootstrap saját CSS-e, illetve az én általam létrehozott *style.css*).

*locales*-ban vannak a lokalizációs fájlok, nyelvi fájlok. Itt akár adhatunk hozzá értelemszerűen új nyelvet, vagy esetlegesen nyelvi javításokat, bővítéseket tehetünk.

*js* mappában vannak a JavaScript fájlok. Mivel itt rengeteg fájl, nagyon sok függvénnyel szerepel, ezért erről mindenképpen szeretnék részletesebben írni, hogy ezáltal könnyebben érthető legyen, amit csináltam, illetve az is, hogy még könnyebben lehessen újrafelhasználni az általam már létrehozott dolgokat.

### JavaScript fájlok

Elsősorban itt vannak a már részletesebben is leírt JavaScript könyvtárak, így itt már csak említésszerűen írnám ezeknek a fájljait: i18next-1.10.3.min.js, jquery-2.1.4.min.js, underscore.js.

A következőekben az általam írt lényegesebb függvényekről említek meg részleteket, hogy későbbiekben akár könnyebben lehessen őket újrafelhasználni, refaktorálni.

*eljarasok.js  
tombRendezettE(t*) – Ellenőrzi a tömb rendezettségét.  
*tombEllenorzes(t)* – Ellenőrzi, hogy a bemenet tényleg tömb-e.  
*tombKiirasa(t, a, b, szoveg)* – Kiír egy tömböt a *tomb* id-jú elembe. Ha a, b egy-egy index, akkor azokat kékkel jelöli. Ha a egy tömb, akkor az a tömbben lévő indexeket jelöli kékkel. Ha b egy tömb, akkor a b tömbben lévő indexeket zölddel jelöli.  
valtozokKiirasa() – Kiírja a *valtozok* id-jú elembe az argumentekben lévő változókat. Egy-egy argumentnek lehet „nev”-e, „ertek”-e és „class”-a amivel a egy CSS class-t tudunk rátenni.   
*barValtoztatasa()* – Attól függően, hogy melyik állapotot mutatjuk, aszerint jeleníti meg a bar-t.  
*lejatszas(k)* – Magától elindítja az algoritmus lejátszását 3 másodpercenkénti léptetéssel.  
*elozoAllas()* / *kovetkezoAllas()* – Lépteti a folyamatot egyik, illetve másik irányba.

*programozasitetelek.js*, *osszetettprogtetelek.js*, *rendezesek.js*, *sor.js*, *verem.js*

Ezek a JS fájlok tartalmazzák az algoritmusokat leíró függvényeket. Felsorolva, hogy melyeket tartalmazzák:

* Programozási tételek: Összegzés, Megszámolás, Maximum-kiválasztás, Keresés, Logaritmikus keresés, Másolás, Kiválogatás, Szétválogatás, Unió, Metszet
* Rendezések: Buborékos rendezés, Minimum-kiválasztásos rendezés, Gyorsrendezés, Összefésüléses rendezés
* Adatszerkezetek és műveleteik: Sor, Verem

## Fejlesztési lehetőségek

A honlapot mivel úgy csináltam meg, hogy van egy nyelvi fájl, ami a szövegeket tartalmazza, ezért könnyen lehet újabb nyelvi fájlokat létrehozni, amivel az egész oldalt le tudjuk fordítani más nyelvekre.

Az általánosan megírt függvények, eljárások miatt könnyen hozhatunk létre, bővíthetjük a gyűjteményt újabb algoritmusokkal, adatszerkezetekkel.

# Tanulási terv

Rengeteg óraterv, tematikus terv létezik, így ezeket itt már nem szeretném részletesebben leírni. Csak vázlatosan említeném meg, hogy az egyes tanórákon miről lenne szó. A tanórákon fontos, hogy a tanuló megismerkedjen, kipróbálja, megértse, majd elsajátítsa az egyes algoritmusok feladatát, működését.

|  |  |
| --- | --- |
|  | Témák |
| 1. óra | Programozási tételek: Összegzés, Megszámolás, Maximum-kiválasztás |
| 1. óra | Programozási tételek: Keresés, Logaritmikus keresés |
| 1. óra | Programozási tételek: Másolás, Kiválogatás, Szétválogatás |
| 1. óra | Halmazműveletek: Unió, Metszet |
| 1. óra | Rendezések: Buborékos rendezés, Minimum-kiválasztásos rendezés |
| 1. óra | Opcionális: Rendezések: Gyorsrendezés, Összefésüléses rendezés |
| 1. óra | Adatszerkezetek: Sor, Verem |
| + 2 óra | Új anyag készítése (részletesebben később) |

## Új anyag készítése

Ami talán érdekessé tehet az ezzel való tanulást, hogy a végén 1-2 tanórában (vagy akár a diákok otthon egyedül is) létrehozhatnak saját algoritmusokat, amiket feltehetnek az oldalra is, így növelhetik az oldal átadott tudást, saját maguk is fejlődnek, illetve mások is megnézhetik a diák által létrehozott részletesen bemutatott tananyagot.

A következőekben lépésenként szeretném megmutatni, hogy általánosságban, hogy hogyan is kell egy új anyagot létrehozni.

1. Először egy minta HTML oldal alapján előállítjuk a vázat. Ebben kicseréljük az algoritmus specifikus nyelvi elemeket a sajátjainkra (vagy akár, ha nem akarunk vele a későbbiekben foglalkozni, beírhatjuk csak a magyar szavakat, kifejezéseket is). Illetve a „Futtatás” gombra egy saját függvényt teszünk rá. Az algoritmusokat megírjuk (pszeudokód, struktogram, C++), és a programsorokhoz melyeknél kiírjuk majd a változók akkori állapotát, ahhoz teszünk nyilakat (egyéni class-okkal), hogy majd a későbbiekben hivatkozva rá meg tudjuk mutatni, hogy az algoritmus lefutása alatt hol tartunk éppen.
2. A *locales* mappában beírjuk az általunk használandó új nyelvi elemeket a JSON fájlba/fájlokba.
3. Ha módosítottunk a HTML-ben egyes elemek kinézetén esetleg új CSS class-t akarunk létrehozni, akkor azt a css/style.css fájlban megtehetjük.
4. Az esemenyek.js fájlban felvesszük gombra felvitt eseményt. Meghívjuk benne a „fuggvenyElottiInit();” nevű függvényt, majd bemenet ellenőrzés után elindítjuk az algoritmus függvényét.
5. A saját algoritmusunknak létrehozhatunk egy új JavaScript fájlt, vagy akár már egy meglévőbe is írhatjuk (attól függően, hogy esetleg kötődik e már egy létező témához). Ebben implementáljuk az új algoritmust, majd az a legjobb, ha minél gyakrabban, minden egyes változásnál kiírjuk a változókat, tömböt/tömböket a „valtozokKiirasa()”, „tombKiirasa()” függvényekkel. A végén pedig meghívjuk a „megjelenitesInit();” nevű függvényt.
6. Ezek után elkészültek a JavaScript fájljaink a gyökérmappában indítsuk el az uglify.sh nevű scriptet, amivel előállítjuk a minify-olt verzióit a fájloknak. Ha nem tudtunk feltenni az UglifyJS 2-t, akkor online is minify-olhatunk (<https://javascript-minifier.com/>), illetve kikerülhetjük úgy is, hogy a HTML fájlunkban, nem ezeket a verziókat hívjuk meg.
7. Az oldal kipróbálása, tesztelése után már csak az marad, hogy feltöltsük a módosításainkat.

## Példa – n faktoriális

Példaként a folyamatot bemutatnám egy algoritmusra, mely kiszámítja az n faktoriálist.

6. Lefuttatom az uglify.sh-t így előállnak a „\*.min.js” fájlok.