  
**Eötvös Lóránd Tudományegyetem**  
Informatikai kar  
Algoritmusok és Alkalmazásaik Tanszék

Turing-gép vizualizálása webes környezetben

Nagy SáraKelemen Márton

mesteroktatóprogramtervező informatikus Bsc  
programtervező matematikus

Budapest 2021

Tartalomjegyzék

[Bevezetés 3](#_Toc60224088)

[1. Elméleti háttér 4](#_Toc60224089)

[Alapfogalmak 4](#_Toc60224090)

[2. fejezet Felhasználói dokumentáció 7](#_Toc60224091)

[A program által megoldott feladat (nem jó ez cím) 7](#_Toc60224092)

[Célközönség (nem jó ez a cím, nem kell ide) 7](#_Toc60224093)

[Rendszerkövetelmények 7](#_Toc60224094)

[Minimális hardveres környezet 7](#_Toc60224095)

[Optimális hardveres környezet 8](#_Toc60224096)

[Szoftveres követelmények 8](#_Toc60224097)

[A szerver-oldali futtatáshoz 8](#_Toc60224098)

[Kliens-oldali futtatáshoz 8](#_Toc60224099)

[Üzembe helyezés 8](#_Toc60224100)

[Hordozhatóság 12](#_Toc60224101)

[A program használata 12](#_Toc60224102)

[Turing-gép létrehozás 12](#_Toc60224103)

[Turing-gép szerkesztő és futtató felület 14](#_Toc60224104)

[Konfigurációs panel 15](#_Toc60224105)

[Gráf diagram 16](#_Toc60224106)

[Hibák és egyéb üzenetek 17](#_Toc60224107)

[3. fejezet Fejlesztői dokumentáció 19](#_Toc60224108)

[A rendszer architektúrája 19](#_Toc60224109)

[Kliens oldal 20](#_Toc60224110)

[Alkalmazott technológiák 20](#_Toc60224111)

[Klens oldali technológiák amiket használok 20](#_Toc60224112)

[Alkalmazott módszerek 21](#_Toc60224113)

[Adatbázis terve 21](#_Toc60224114)

[Modul és osztályszerkezet 21](#_Toc60224115)

[A Kliens-oldal osztályszerkezet 21](#_Toc60224116)

[A felhasználó felület terve 21](#_Toc60224117)

[Telepítés fejlesztői célokra 21](#_Toc60224118)

[Megvalósítás 21](#_Toc60224119)

[4. fejezet 21](#_Toc60224120)

[5. fejezet Tesztelés 22](#_Toc60224121)

[6. fejezet Irodalomjegyzék 23](#_Toc60224122)

# Bevezetés

Alan Turing angol matematikus a Turing-gép fogalmát 1936-ban dolgozta ki a matematikai számítási eljárások, valamint algoritmusok pontos leírására, illetve számos automatikusan is megoldható kalkulációra. Turing alkotta meg az akkortájt még nem létező számítógépek tevékenységének modellezésére. Egyetemi tanulmányaim alatt felettébb felkeltette érdeklődésemet a téma, emiatt döntöttem emellett, mivel úgy vélem napjainkban is fontos részét képezi az informatikai fejlődés világának. Korábbi tanulmányaim során én is találkoztam már ezzel az algoritmussal, és akkoriban nem volt még ilyen oktatói program. Ezért alkottam meg ezt a programot, hogy más diákok számára könnyebben megérthető legyen a tananyag.

Szakdolgozatom célja, hogy mások számára vizuális eszközökkel könnyebben megérthető legyen a Turing-gép működése. A szakdolgozat két fő részből épül fel, melyek a következőek: az algoritmus implementálása egy programozási nyelvre, valamint a másik pedig a szemléltetés kivitelezése. A modell egyetlen programot hajt végre bármilyen inputra (ami szalagon érkezik), azaz tekinthető egy célszámítógépnek. A gép főbb részei: a vezérlőegység (CPU), a szalag, amely az inputot hivatott megvalósítani és egy író-olvasó fej, amely a szalagot még léptetni is tudja. A megvalósítandó gép determinisztikus, továbbá minden esetben definiált az átmenet. A végtelen szalag potenciálisan végtelen tár.

A lehetséges Turing-gépek egy előre definiált fájlformátumban adhatók meg. Ezek közül választva futtathatunk többféle gépet is. Egy-egy futtatásánál választható több mód is: lehetséges az eredmény azonnali megtekintése, illetve a futás végig nézhető lassítva és lépésenként, kézileg léptetve is.

# Elméleti háttér

## Alapfogalmak

Forrás: https://web.cs.elte.hu/~tichlerk/logika/h/tg.pdf

A Turing-gép egy olyan M = {Q, Σ, Γ, δ, q0, qi, qn} rendszer, ahol

* Q az állapotok véges, nemüres halmaza,
* q0, qi, qn ∈ Q, q0 a kezdő- qi az elfogadó- és qn az elutasító állapot,
* Σ és Γ ´ábécék, a bemenő jelek illetve a szalagszimbólumok ábécéje úgy, hogy Σ ⊆ Γ és ⊔ ∈ Γ \ Σ.
* δ : (Q \ {qi, qn}) × Γ -> Q × Γ × {L, R, S} az átmenet függvény.

• A Turing-gép működésének fázisait a gép konfigurációival írjuk le. A Turing-gép konfigurációja egy uqv szó, ahol q ∈ Q és u, v ∈ Γ∗, v != *ε*.  
A konfiguráció a gép azon állapotát tükrözi, amikor a szalag tartalma uv (uv előtt és után a szalagon már csak ⊔ van), a gép a q állapotban van, és a gép író-olvasó feje a v szó első betűjén áll.  
• A gép kezdőkonfigurációja egy olyan q0u szó, ahol u csak Σ-beli betűket tartalmaz.  
  
• Egy Turing-gép konfigurációátmenetét az alábbiak szerint definiáljuk. Legyen uqav egy konfiguráció, ahol a ∈ Γ, u, v ∈ Γ∗.

* Ha δ(q; a) = (r; b; R), akkor uqav |- ubrv’ , ahol v’ = v , ha v != *ε*, különben v’ = ⊔,
* ha δ(q; a) = (r; b; S), akkor uqav |- urbv,
* ha δ(q; a) = (r; b; L), akkor uqav |- u’rcbv, ahol c ∈ Γ és u’c = u, ha u != *ε*, különben u’ =u *és* c = ⊔ .

• Azt mondjuk, hogy M véges sok lépésben eljut a C konfigurációból a C’ konfigurációba (jele C |-\* C’), ha van olyan n ≥ 1 és C1,….., Cn konfigurációsorozat, hogy C1 = C, Cn = C’ és minden 1 ≤ i < n-re, Ci |- Ci+1.

• Ha q 2 {qi, qn}, akkor azt mondjuk, hogy az uqv konfiguráció egy megállási konfiguráció. q = qi esetében elfogadó, míg q = qn esetében elutasító konfigurációról beszélünk.

• Az M által felismert nyelv (amit L(M)-mel jelölünk) azoknak az u ∈ Σ∗ szavaknak a halmaza, melyekre igaz, hogy q0u |-∗ xqiy valamely x, y ∈ Γ∗, y != *ε*  szavakra.

• Egy L ⊆ Σ∗ nyelv Turing-felismerhető, ha L = L(M) valamely M Turing-gépre. Továbbá, egy L ⊆ *Σ*∗ nyelv eldönthető, ha létezik olyan M Turing-gép, mely minden bemeneten megállási konfigurációba jut és felismeri az L-et. A Turing-felismerhető nyelveket szokás rekurzívan felsorolhatónak, az eldönthető nyelveket pedig rekurzívnak is nevezni. A rekurzívan felsorolható nyelvek osztályát RE -vel, a rekurzív nyelvek osztályát pedig R-rel jelöljük.  
• Tekintsünk egy M = [Q, Σ, Γ, δ, q0, qi, qn] Turing-gépet és annak egy u ∈ Σ∗ bemenő szavát. Azt mondjuk, hogy M futási ideje (időigénye) az u szón n (n ≥ 0), ha M a q0u kezdőkonfigurációból n lépésben (konfigurációátmenettel) jut el megállási konfigurációba. Ha nincs ilyen szám, akkor M futási ideje az u-n végtelen.

• Legyen f : N -> N egy függvény. Azt mondjuk, hogy M időigénye f(n) (vagy, hogy M egy f(n) időkorlátos gép), ha minden u ∈ Σ∗ input szóra, M időigénye az u szón legfeljebb f(|u|).

• A **k**-szalagos Turing-gép egy olyan M = [Q, Σ, Γ, δ, q0, qi, qn] rendszer, ahol

* Q az állapotok véges, nemüres halmaza,
* q0; qi; qn ∈ Q, q0 a kezdő- qi az elfogadó- és qn az elutasító állapot,
* Σ és Γ ábécék, a bemenő jelek illetve a szalagszimbólumok ábécéje úgy, hogy Σ ⊆ Γ és ⊔ ∈ Γ \ Σ,
* δ : (Q \ {qi, qn}) × Γk -> Q × Γk × {L, R, S}k az átmenet függvény.

• A **k szalagos Turing-gép** konfigurációja egy szó, ahol q ∈ Q és ui, vi ∈ Γ∗, vi != *ε* (*1* ≤ i ≤ k). Az u szóhoz tartozó kezdőkonfiguráció: ui = *ε* (1 ≤ i ≤ k), v1 = u, és vi = ⊔ (2 ≤ i ≤ k). Időigény: mint az egyszalagosnál (konfigurációátmenetek száma alapján).

• Szófüggvényt kiszámító Turing-gép:

Az M (determinisztikus) Turing-gép kiszámítja az f : Σ∗ -> Γ∗ szófüggvényt, ha M minden u ∈ Σ∗-ra olyan vqw megállási konfigurációba jut (q ∈ {qi, qn}), ahol vw = f(u) (szóeleji és szóvégi ⊔-ektől eltekintve). Időigény: mint fent (konfigurációátmenetek száma alapján)

# Felhasználói dokumentáció

## A program által megoldott feladat (nem jó ez cím)

A terjedelmes tananyag tanulása során néha kevés idő jut arra, hogy egy  
összetettebb algoritmust képesek legyünk teljes egészében megérteni. Jelen oktatói program ebben próbál segítséget nyújtani, hogy komplexebb algoritmusokkal tudjon az ember dolgozni. A gyakorlaton alapul vett gráfos ábrázolással, illetve szalagbemenettel fogja majd illusztrálni a program az algoritmus működését. Ezzel hatékonyabbá válhat a tanulás, valamint például segíthet leellenőrizni az elkészített házifeladat helyességét is. Emellett az órán az oktatónak nem muszáj feltétlen minden alkalommal rajzolni is a táblára, helyette csak kivetíthető a webes alkalmazás.

## Célközönség (nem jó ez a cím, nem kell ide)

A program elsősorban informatikai karon lehet hasznosítható, azon belül pedig a formális nyelvek, algoritmusok, számításelméletet oktató vagy tanuló egyetemi tanárok és hallgatóik számára készült. Ezt mutatja a program weboldal formátuma is: minimális hardverkövetelménye okán nincs szükség komolyabb eszközre, így szinte bármilyen intézményben könnyedén alkalmazni lehet. A program segítséget nyújthat a hallgatóknak az egyes előre betáplált példa feladatok  
kipróbálásában, valamint ezek újra szerkesztésében. Továbbá segít az oktatóknak is például a zárthelyi feladatok összeállításában, vagy a diákok által megírt dolgozatok kijavításában.

## Rendszerkövetelmények

### Minimális hardveres környezet

Úgy vélem, hogy legegyszerűbben személyi számítógépen érdemes használni a programot. A piacon legkedvezőbb áron elérhető ilyen eszköz a Rasperry Pi 4 modell B miniszámítógép.

* 1,5 GHz (1 500 MHz)
* 4 GB memória

Már ez az eszköz is képes futtatni egy web böngészőt, valamint egy webszervert is. Tehát elmondható, hogy minimális hardveres követelményei vannak a programnak.

### Optimális hardveres környezet

Szerver számítógép. Minta a http://webprogramozas.inf.elte.hu/. Egy ilyen szerver képes egyszerre több felhasználót kiszolgálni, ha mondjuk egy tanteremben vagy előadáson a tanár kérésére minden diáknak ki kell próbálni a programot.

### Szoftveres követelmények

#### A szerver-oldali futtatáshoz

Személyi számítógépen apache web szervert érdemes telepíteni. Ilyet legegyszerűbben a XAMPP nevű szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomagot érdemes installálni. Egyszerű és bármely desktop operációs rendszerre jó lehet. Továbbá, szerver számítógépre más eljárással érdemes telepíteni webszervert, itt érdemes a szerver üzemeltetők segítségét kérni.

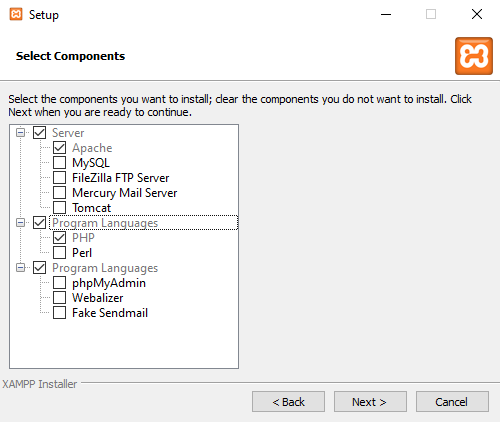
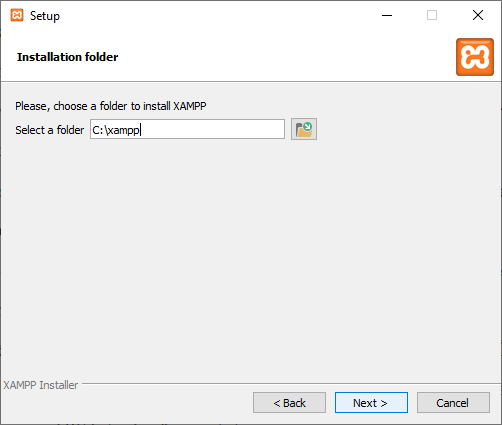
Kliens-oldali futtatáshoz

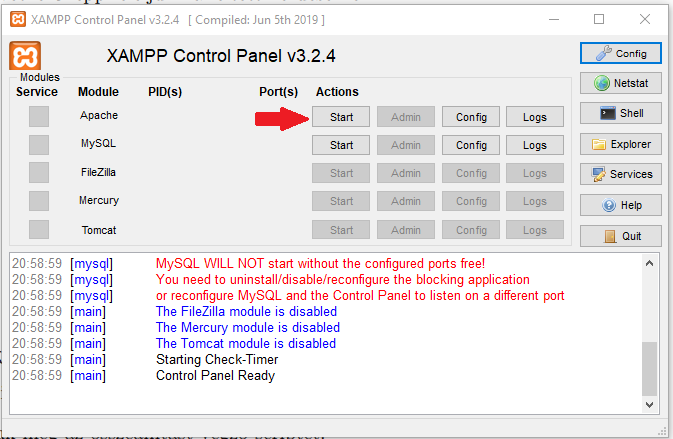
ES6 és CSS3 támogatású webböngészők:

* Desktopon: legalább az alábbi kiadásokon: Firefox 24, Chrome 37, Safari 9, Edge 12, Opera 24
* Mobil, tablet eszközökön: IOS safari 10, Android Browser 81

## Üzembe helyezés

Ha **személyi számítógépről** szeretnénk hostolni a programot, akkor szükség van egy webszerver szoftverre. Én a XAMPP-t tudom ajánlani én is ezt használtam a fejlesztés során.

* https://www.apachefriends.org/hu/download.html (a link 2020.12.23-án érvényben lévő) töltsük le az operációs rendszerünkhöz a megfelelőt illetve PHP 8-as verziót tartalmazzon.
* Telepítéskor lehet választani a szoftver csomag mely részei legyenek csak telepítve, ekkor csak az Apache web szervert pipáljuk be, másra nincs szükségünk.
* Ezután kérdezni fogja hol legyen a XAMPP telepítési helye, itt érdemes az operációs rendszerrel azonos lemezre telepíteni annak is főkönyvtárjába.  
  
* Ha a telepítés végbement, akkor lehet máris indítani a webszervert. Ehhez meg kell keresni a „xampp-control.exe” indító alkalmazást. Erre kattintva jelenik meg a control panel, ahol az Apache web szervernek a start gombjára kell rá kattintani.

2.sz. ábra:  


* Ahhoz, hogy lefordítódjanak a PHP fájlok, el is kell őket helyezni egy megfelelő könyvtárban. Telepítés során, ha az általam ajánlott könyvtárat vesszük, akkor annak egy alkönyvtárába kell tenni, ami ezen az elérési úton van. C:\xampp\htdocs

Továbbá, ha **szerver számítógépről** szeretnénk több felhasználónak elérhetővé tenni a programot, akkor ellenőrizni kell rajta, van-e már telepítve *PHP* fordító. Ha nincs, akkor érdemes a rendszer karbantartóját segítségül hívni.

**Kliens oldalon,** ha a felhasználó szeretné használni a programot, akkor csak egy böngészőre van szüksége, ez nem jelenthet nagyobb gondot, legtöbb operációs rendszerrel ellátott eszközön alapvetően van. Ha személy számítógépről történik a hostolás, akkor az apache szervert indítva, a böngésző címsorába „localhost/szakdolgozat” -t kell begépelni. A „szakdolgozat” szót arra kell kicserélni, amilyen a „C:\xampp\htdocs” alatt található alkönyvtár neve.

Ha szerver számítógépen van futtatva a szakdolgozati program, akkor egy URL link segítségével bármilyen webböngészővel ellátott eszközzel el lehet érni a programot, feltéve, ha nincs jogosultsághoz kötve a szerver látogatása.

### Hordozhatóság

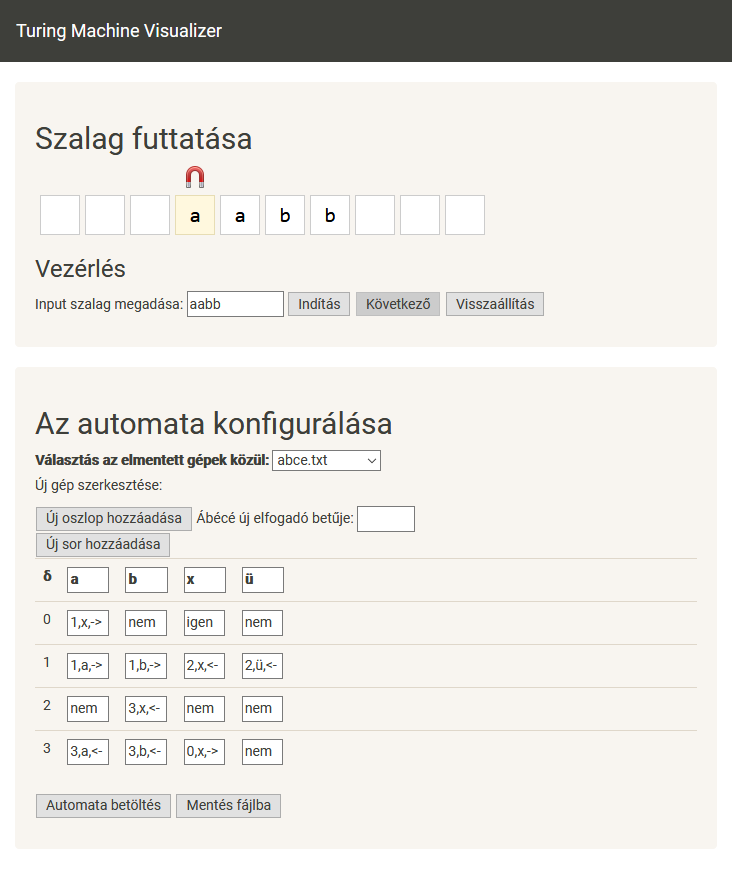
A program egyszerűen költöztethető. Ellenőrizni kell futtatható-e az új környezetben PHP fájl, illetve még milyen régi a fordító program. Ha ezek teljesülnek, akkor a program forrásfájlokat egyszerűen át lehet másolni egyik helyről a másikra. Semmilyen egyéb más teendő nincs.

## A program használata

### Turing-gép létrehozás

Személyi számítógépről hostolva meg kell bizonyosodni, hogy fut-e a webszerver a háttérben, csak ezután indítható a program. Ha pedig webszerverről akarjuk elérni, akkor bármilyen internetezésre alkalmas eszközzel, például: egy telefon vagy tabletnek a webböngészőjébe kell begépelnie megfelelő URL-t, ami a programra mutat. Ez lehet localhost vagy lehet egy szokványos internetre mutató link. Ha megfelelő az URL akkor az alábbi [ÁBRA] ablakával találkozik. A program 2 fő részből áll. Kezdetben ezek csak részben láthatók. További elemek a program futása során kerülnek csak elő.  
 Csak akkor indítható, ha a két komponens is készen áll. Ugyanis egyik komponens a másik komponens nélkül nem adnak elég adatot ahhoz, hogy a gép el tudjon indulni. Első a szalag bediktálás, második a gép konfigurációjának megadása. Ha a felhasználó rátekint az oldal felépítésére, akkor azt látja, hogy a szalag vezérlő elem az oldal tetején helyezkedik el, míg a konfigurációs panel az oldal alján. A két panel sorrendben való kitöltésére nincs szabály megadva, egyetlen lényeges szempont van, hogy minden beviteli mező ki kell legyen töltve.

x.sz ábra:



### Turing-gép szerkesztő és futtató felület

Az itt látható egyik főkomponens [ÁBRA] segítségével lehet többek között megadni a szalagot, annak működését vezérelni, illetve animálva lehet látni az olvasó fejet, ahogy olvassa a szalag betűit. A megadni kívánt input szalagot az (*Input szalag megadása*) mezőbe bele kell kattintani egérrel, és be kell gépelni egy szót, ami csak betűket tartalmaz, nem lehet ékezetes, nem lehet nagybetű és szám sem lehet. Az angol ábécé betűit fogadja csak el. Az (*Input szalag megadása*) mező megfelelő kitöltés után lehet csak az indítás gombra kattintani. Ha a másik -fő komponens- konfiguráció is megfelelően van kitöltve, akkor indul csak el az automata. Az adatok betáplálása és indítása után, a működő gépet nem lehet szerkeszteni, csak ha végbement (*Terminált*). Lehetséges megállítani a működésben lévő automatát, lehet léptetni, tekerni, valamint újraindítani is.

x.sz. ábra:



### Konfigurációs panel

Itt látható a program másik fő komponense [ÁBRA]. Ezen a kis felületen rengeteg funkció lett összeállítva. Elsőként ki lehet választani már előre elkészített példa konfigurációkat. Ezeket később átszerkesztve el lehet menteni, valamint bármikor újra lehet beimportálni.  
Ha a felhasználó nem szeretne régebbi beálltásokat alkalmazni, akkor saját maga is létrehozhat egyet.

Ezt az alábbi táblázatban teheti meg. A táblázatot szabadon lehet szerkeszteni, bármelyik cellát felül lehet írni, illetve új sorokat és oszlopokat lehet hozzáadni. Új oszlopokat csak akkor lehet hozzáadni, ha kötelezően megadunk egy elfogadó betűt, ami a táblázat fejlécébe helyeződik. Új sornál nem kell ilyesmit tenni, ott automatikusan generálódik egy új szám a táblázat első oszlopába, ez megegyezik az állapotok számával, figyelve a korábbiakat, hogy ne legyen duplikáció, illetve, hogy növekvő sorrend legyen.

Ha a felhasználó törölni szeretne az oszlopok vagy a sorokból, akkor az egér segítségével sor esetén a sor elejére kell kattintani, oszlop esetén az oszlop fejlécére kell kattintani, ekkor mindkét esetben a sor vagy az oszlop törlésre kerül a táblázatból.

A kész konfigurációt szükség esetén elmenteni is lehet egy fájlba (*Mentés fájlba*), amit később újra be lehet importálni.

Mindezek után a Gépbe be lehet tölteni az adatokat az (Automata betöltése) gombbal.

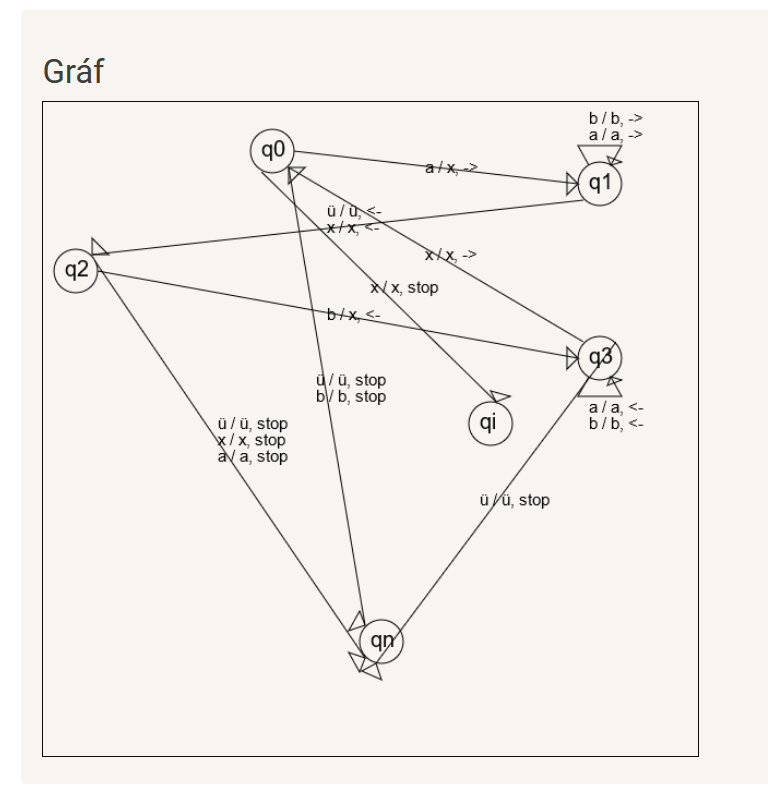
x. sz. ábra:



* + 1. Gráf diagram

Az automata helyes betöltése után, egy új panel nyílik meg a konfigurációs panel alatt. Itt kerülnek kirajzolásra a gráf diagram elemei, ahogy az ábra is mutatja [ÁBRA] Ezek leggyakrabban a beállítástól függően, állapotokat és átmeneteket jelölnek. Az állapotok gömbök képében jelennek meg. Míg az átmenetek vonallal, nyíllal a végén jelezve a haladási irányt rajzolódik ki a gömbök között. Minden átmenet rendelkezik utasítással. Ezek szövegesen, külön-külön az átmenetek fölé kerülnek kirajzolásra.

x.sz. ábra:



* + 1. Hibák és egyéb üzenetek

Adott esetben megtörténhet, hogy a felhasználó szembesül valamilyen hibajelenséggel. Túlságosan behatóan nem kell ismernie ezeket, azonban nem árt néhány jelenséggel tisztában lennie. Ezek pedig a következőek:

* *„Probléma merült fel az oldal betöltése közben” 🡪* Több dolog okozhatja ezt. Leginkább az internet elérésével lehetnek gondok. Ekkor érdemes ellenőrizni az internet kapcsolatot az eszközön, amivel próbáljuk futtatni a programot.
* *„Hiba történt a fájlok importálása közben” 🡪* Megeshet, hogy kész munkánk elveszhet például vírusirtó által, vagy a program átmozgatása közben. Mindenesetre, a forrásfájlok mellett megtalálható egy mappa, ahol az elmentett gépi kunfigurációk találhatók, minden indítás előtt érdemes ellenőrizni ezt a könyvtárat.
* *„Futás közbeni hibaüzenetek”🡪* Minden hiba, amit a program jelezhet, szöveges formában jelenik meg. Általában felugró szövegdoboz formájában. Ilyen hibák lehetnek: rosszul megadott paraméterek, nem megfelelően kitöltött szövegbeviteli mezők. Érdemes ezeket újból alaposabban átvizsgálni, hogy vajon mit rontottunk el.

# Fejlesztői dokumentáció

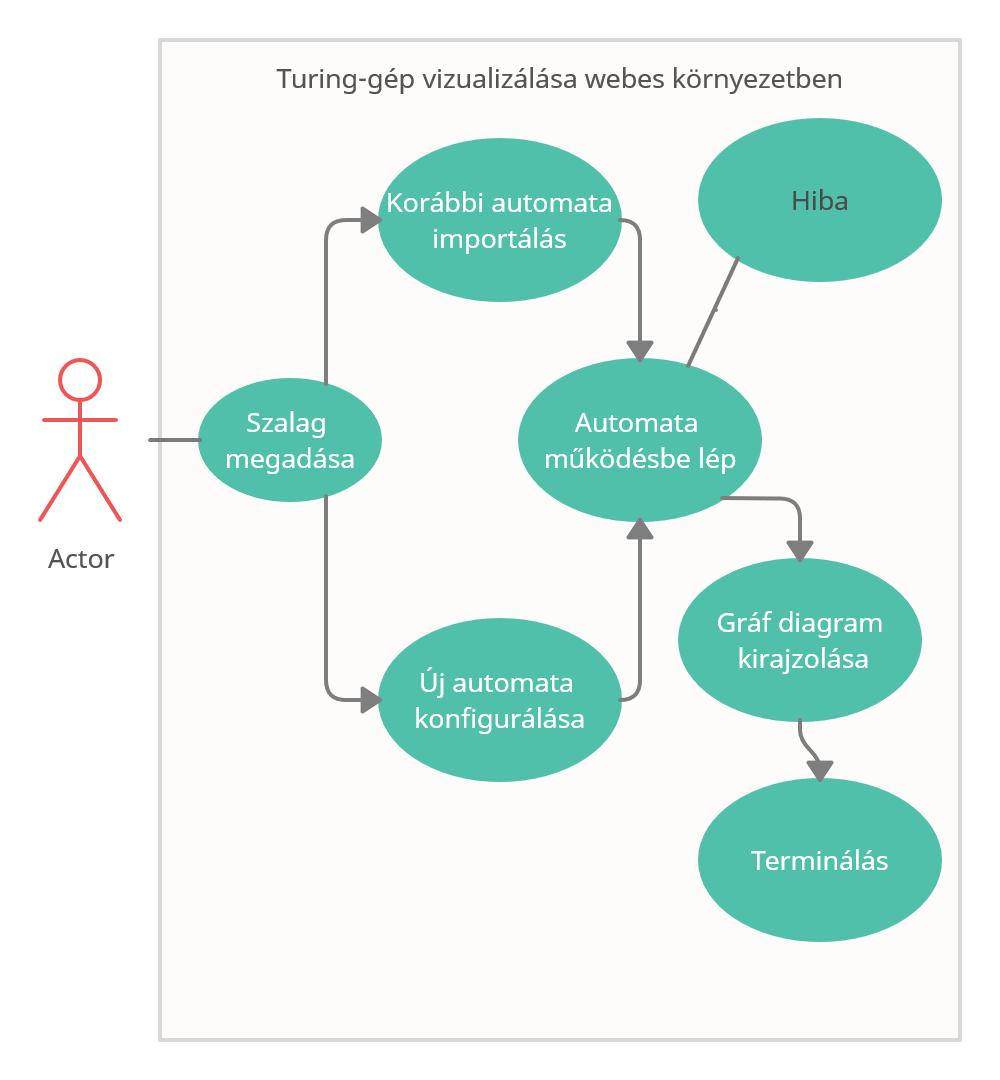
## Tervezés

A tervezési fázis során, oktatásra megfelelő program megalkotása volt a célkitűzésem. Fontos szempont volt, hogy a program platform független, hordozható és nagyon egyszerű üzemeltetésű legyen. Ezért választottam a webes technológiát mert bármilyen internetezésre használható eszközön is be lehet mutatni az alkalmazást. Továbbá fontos volt még az is, hogy olyan technológiákat használjak a fejlesztés során, amelyet kötelezően tanítanak az egyetemen, így akár könnyedén továbbfejleszthetővé válik a kód.

## Felhasználói esetek

A következő oldalon lesz megtalálható a Use case diagram, mellyel bemutatásra kerül az alkalmazás funkciói. Az alkalmazásnak egy eszközön futtatva egy felhasználója lehet. A program két fő részből áll. A szalag importálása és annak működésközbeni léptetése, másik fő komponens pedig az automata konfigurálása, ahol megadhatjuk miképpen működjön a gép.

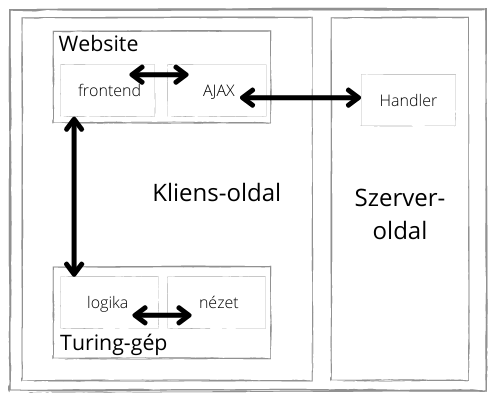
x.sz. ábra:



## A rendszer architektúrája

Az eredeti elgondolásnak megfelelően, **kliens-szerver** architektúra szerint alakítottam ki a program szerkezetét. Kód mennyiségének tekintetében, a kliens oldalra fordítottam több erőforrást. Ezt eredetileg is így terveztem, inkább a felhasználó barátiabb megjelenésre tettem a hangsúlyt.

x.sz ábra:



A kliens a szerverrel HTTP protokollon keresztül kommunikálnak, AJAX segítségével. A szerver egyedül a fájlkezeléssel foglalkozik. Feladata beolvasni a fájlt, azt feldolgozni és tovább küldeni a kliens oldalra. Ha a kliens oldal elvégezte az adatok feldolgozását, akkor azt elküldi a szerver felé fájlba mentés céljából. A kliens oldalon történik a gép logikája, megjelenítése, és a felhasználó interakciók kezelése.

### Kliens oldal

* **Frontend:** a weboldal megjelenítésért felel. A felhasználó itt lép kapcsolatba a programmal. Minden felhasználói cselekvés itt kerül lekezelésre. Minden a fájlokkal kapcsolatos művelet, frontend oldalról az AJAX modul segítségével kerül a szerver oldalra.
* **Ajax modul:** kapcsolatot teremt a szerver és kliens oldalakkal. Nagy előnye, hogy frontend oldal frissítése a háttérben történik, nem tölti be újra az egész oldalt. Ettől ergonomikusabbá válik a felhasználó számára, másrészről egy jól használható technológia kifejezetten a két oldali kommunikáció lebonyolítására.
* **Turing-gép futtatása és konfigurálása:** a program egészéhez képest egy nagy részét fedi le. Egy nézetből és egy logikai egységből áll. A belső logika állapota kerül megjelenítésre nézet által, illetve a nézet kezeli a felhasználó iterációit, amely a logikának kerül átadásra.

## Alkalmazott technológiák

Kliens oldal egy weboldal, melynek elengedhetetlen programnyelve a HTML, de ettől még csak statikus az oldal. Dinamikus használatához Javascript objektumorientált szkript programozási nyelvet használok. Logika és nézet használatához, illetve megvalósításához elengedhetetlen, hogy a weboldal dinamikusan kövesse a felhasználói utasításokat. Ezen kívül, esztétikailag szebb megjelenés érdekében használok Bootsrapt és CSSt, illetve a Javascpriten belül jQuery könytárat és AJAX modult használok.

Szerver oldalon PHP 7-es verzióját használom. Ennek fordítása a hosztoló gépen történik és nem a kliens web böngészőjében. Emiatt a hosztoló gépen kell lennie PHP 7 fordítónak.

#### Klens oldali technológiák amiket használtam

* **HTML:** weboldal leíró nyelv, szabványosított. 5-ös, legújabb verzióját használtam, számos újítása van, aminek köszönhetően kevesebb szkriptet kellett írnom.
* **CSS:** ugyancsak szabványosított nyelv. Weboldalak stílusát vele leírni. Programozási nyelvnek nem lehet mondani, inkább egy statikus kódolásnak lehet nevezni, amit a W3C szabvány alapján lehet használni. A webböngésző fordítja le úgy, ahogy a HTML-t vagy JavaScriptet.
* **JavaScript:** objektumorientált szkript nyelv. Ez a nyelv nem szabványosított, legismertebb böngészők különböző fordítóprogramokat használnak rá. Én fejlesztés során a web böngészőnek Mozilla Firefoxot használtam, ami SpiderMonkey fordító motort használ.
* **Bootstrap:** JavaScript és CSS keretrendszer. Segítségével az oldalt könnyebben stílusosabbá lehet alakítani. Legnagyobb előnye, hogy reszponzívvá teszi a weboldal felületet, azaz követi a webböngésző méretét, ami gyakran megegyezik a készülék méretével. Pl.: telefon, tablet kijelző
* **jQuery:** JavaScript keretrendszer, egyszerűbbé teszi a Selectorok használatát, CSS és DOM navigációt, AJAX kezelést teszi egyszerűbbé.
* **AJAX:** aszinkron interaktív programozási nyelv. JavaScript kiegészítésére alkalmas, ha a kliens és szerver kapcsolatát szeretnénk kivitelezni. Kis mennyiségű adatcserénél érdemes használni, mert az oldal teljes újra töltése nélkül lehet az oldal elemeit frissíteni.

#### Szerveroldali technológiák, amiket használtam

* **PHP7:** Azért választottam ezt a technológiát, mert az alapképzésen a törzs tárgyak között ezt tanították nekünk. A legismertebb szerveroldali szkript nyelv, sajnos az idő kissé eljárt felette, más programozási nyelvek is alkalmasak már hasonló feladatok ellátására. Ami előnye, hogy a régi létezése okán nagy a support közösség, ezáltal, ha elakadtam egy kérdésben, könnyedén találtam rá megoldást az interneten.

## Alkalmazott módszerek

MV (Model-View) architektúra

Turing gép vezérlőnél használom

* A **model** tárolja az állapotot, ami hatással van a kapcsolódó műveletekre, algoritmusokra. Az állapotváltozás befolyásolja ezek működését.
* A **view** rajzolja ki modell állapotát. Kezeli a felhasználói interakciókat és azokat átadja a modell számára feldolgozásra.

## A Kliens-oldal forrás elemei

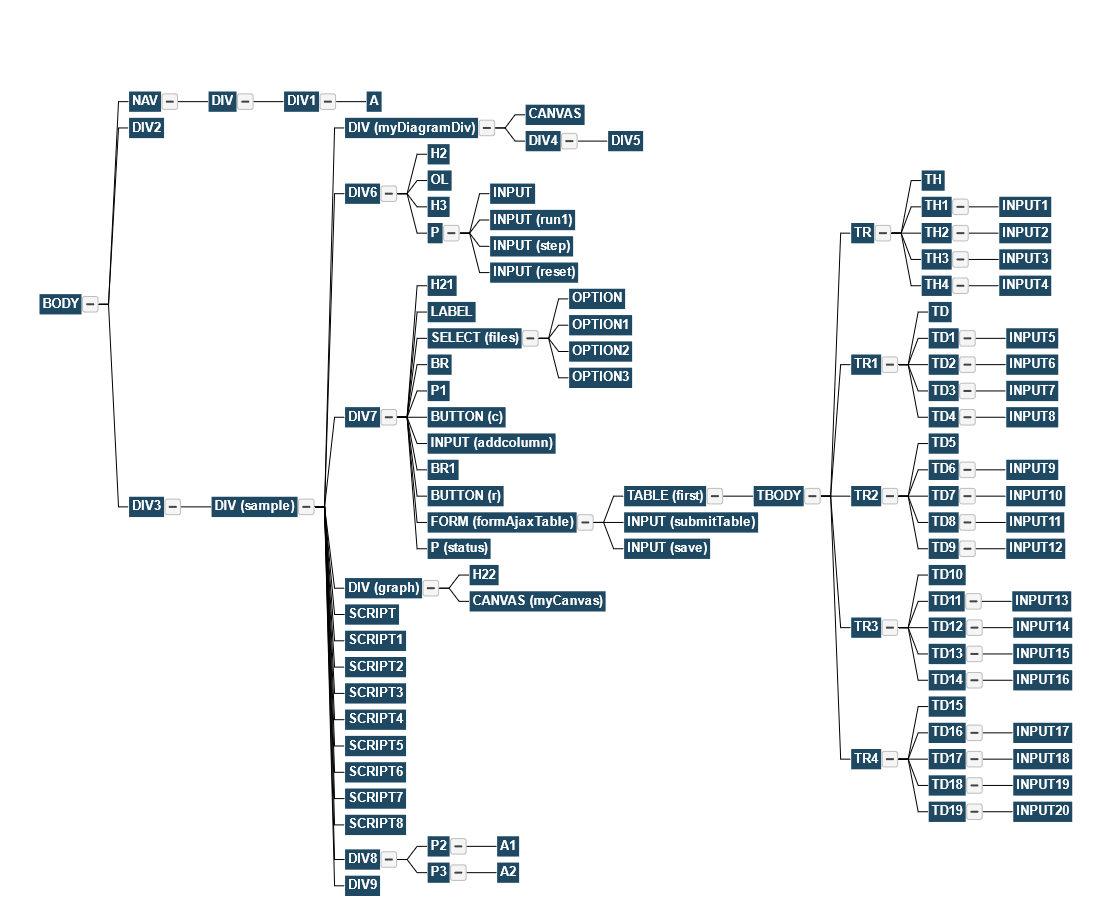
* Index.html / Főoldal
* js.js / Kliensoldali logikért felel
* turing.js / Könyvtár
* turingjs.js / Turing-gép modulért fele
* style.css / Az oldal stílusjegyei itt kerültek implementálásra
* graph.js / Grafikus állapot diagram rajzolás

### Index.html

#### Dom szerkezet

Az alábbi [Ábrán] látható az Index.html szabványban használt *tag*jei illetve a Javascript világból ismert DOM elemek összekapcsolt fa szerkezetben. Mivel a fejlesztésem során nem objektumorientáltan programoztam, ezért osztályszerkezeteket nem tudok megjeleníteni, helyette a főoldalam dinamikus struktúráját tudom bemutatni.

x.sz. ábra:



#### Részletes funkció leírás

Index.html a kezdőoldal a felhasználó számára. Mint html fájl itt kerül beimportálásra minden stíluselem. Ezek mind a <head> tagben kerülnek bele. Itt importálom be a bootstrap stílus osztályt, innentől kezdve az egész oldalra érvényes lesz a benne található osztályok. Ezen kívül szükségem volt még néhány stílusjegyre, amivel az oldal szebb és érthetőbbé vált. Ezért beimportáltam a style.css fájt, ami mindezt tartalmazza.

A <body> tagben található meg a program 2 fő része. Az első, ahol a szalagot tudjuk betáplálni, itt egy <input text> tag van, ami szöveget vár. Csak az angol ábécé betűit lehet használni, illetve csak kis betűt, ezen kívül számokat nem fogad el. A text tag mellett ott helyezkedik el három gomb melyekkel vezérelni lehet a szalag működését. Az indítás gombbal indul el a szalag olvasása. A szalag olvasásának két állapota van, az első, amikor megy, második mikor áll az olvasó. Ezen kívül az olvasó fej képes automata módon olvasni, megállni csak ha szalagvégéhez ér. Vagy van a másik mozgása az olvasónak mikor kézzel tudjuk léptetni betűnkként. Ekkor minden gombnyomásra olvas egy betűt és megáll. Továbbá, a gombok felett helyezkedik maga a szalag. Indítás előtt csak egy üres szalag látszik, indítás után kerül a szalagra a szó, amit megadtunk a text input mezőben. Indítás után a szalag animálása nem változik, kizárólag az olvasó fej mozog az input szöveg felett és alatt, adott esetben a szöveg változik.

Tovább haladva a lapon, megérkezünk a konfigurációs panelhez. Itt az első elem egy *drop down list,* amely kilistázza mely korábbi turing-gépek találhatóak a megadott elérési úton. Egyet közülük kiválasztva megkapjuk annak tartalmát és lehet is betölteni az automatába indulásra készen. Ellenben, ha nem szeretnénk lementett fájlokat felhasználni, akkor készíthetünk új konfigurációt. Ehhez alkottam egy táblázatot, amely minden cellája szerkeszthető, kivéve az állapotok sorszáma, mert azokra nincs szükség, hogy a felhasználó módosítsa. Ha törölni szeretnénk bármely sort vagy oszlopot, akkor a bal egér kattintással rá kell kattintanunk a kívánt oszlop tetejére, vagy sor esetén, a kívánt sor elejére. Oszlopra kattintásnál vigyáznunk kell, mert a cellák input mezőkből állnak, melyekre, ha rá kattintunk, akkor a cella tartamot módosítjuk és nem az oszlopot töröljük. Ezért segítségül létrehoztam egy segítő lebegő szöveget, ez akkor jelenik meg, ha az egér a cellák felett van. Bármely cella fölé ér az egér, megjelenik az egér mellett a cella sorszáma, illetve, ha törlés funkcióra megyünk, akkor az egér mellett megjelenik az üzenet, hogy itt erre a felületre való kattintással törlés történik meg. Ha befejeztük a konfigurálást, akkor két út van előttünk. Elmenthetjük egy fájlba, és utána töltjük be az automatába vagy nem mentjük el és mentés nélkül kerül betöltésre.

Betöltés után a második főpanel alatt megjelenik az állapot diagramm. Itt a konfigurálási adatokból rajzolódik ki egy gráf. Egyszer történik meg, automata betöltéskor és utána egy állapotban marad.

## Js.js

Ez a szkript fájl tartalmazza a főoldal Dom elemeinek kezelését, egyet kivéve, az állapot diagrammét. Ott annyira összetett rajzolási formát kellett kialakítani, hogy muszáj volt más fájlba tenni.

### Adattagjai:

* *myForm* 🡪 Egy olyan adattag, amelynek értékül egy Dom objektumot adtam, mivel sokat fogom hívni, ezért volt szükség eltárolni. Egy ajax kéréshez használom, amivel formot küldök feldolgozásra.
* *myFile 🡪* Egy olyan adattag, amelynek értékül egy Dom objektumot adtam, mivel sokat fogom hívni, ezért volt szükség eltárolni. Egy ajax kéréshez használom, amivel fájlt küldök feldolgozásra.
* *statusP 🡪* Egy olyan adattag, amelynek értékül egy Dom objektumot adtam, mivel sokat fogom felhasználni. Az upload.php-től jövő karater láncot íratom ki vele.
* *finalMap 🡪* Map az egy map objektum, amiben kulcs-értékpárként tárolom el a konfigurációs beállításokat. Ezt a Mapet adom át többek között az állapot diagram kirajzolónak.

### Metódusai:

* myForm.onsubmit = function(event) {} :void 🡪 Egy eventet váró függvény, mely egy dom eseménykor kerül meghívásra. Ez a függvény adja át a fájlt feldolgozásra az Upload.phpnak ajax kéréssel. Ha sikerült a feldolgozás, akkor a fájl tartalma egyből visszakerül ebbe a függvénybe.
* function removeColumn(e){} :void 🡪 Egy eventet váró függvény, amely a kapott paraméterek alapján törli a log-ából a megfelelő oszlopot, amit a bemeneten kapott.
* function reNumber(table, row, i) {}:void 🡪 Egyik legfontosabb függvény itt. Paraméterül táblázatot, sort és indexet kap. Ha bármilyen módosítás történik a táblázatban, azaz cella törlés vagy hozzáadás, akkor újra számolja a táblázat sorait, oszlopait és frissíti azt a nézetben.
* function addCell(currentRow, rownr) {}:void 🡪 Akkor hívódig meg, ha új oszlop kerül hozzáadásra. Ekkor tudni kell, hogy mely oszlopnak kell adni cellát.
* function addColumnHeader(headerRow) {}:void 🡪 Táblázat fejlécét szerkeszti valaki, például egy új oszlop esetében, akkor ez a függvény helyezi megfelelő helyre az új oszlop nevet.
* function addColumn(){}: void 🡪 Gombnyomásra hívódik meg, mikor új oszlopot szeretnék hozzáadni, ez a függvény rendezi megfelelő helyre az új oszlopot.
* function addRow(e) {}:void 🡪 Egy eventet vár. Ha sort szeretnénk a táblázathoz adni, akkor a gombot lenyomva ez a függvény adja hozzá.
* function removeRow() {}: void 🡪 Egy kiválasztott sornak a törlését végzi el. Ha utolsó sor a táblázatban, akkor azt nem lehet törölni, figyelmeztetést ad.
* function removeColumn(e){} {}: void 🡪 Egy kiválasztott oszlop törlését végzi el. Ha utolsó oszlop a táblázatban, akkor azt nem lehet törölni, figyelmeztetést ad.
* function getByValue(map, searchValue) {}:string 🡪 Ez egy segédfüggvény, mellyel egy adott mapben tudok érték alapján keresni, vissza tudja adni annak az értéknek a kulcsát.
* myForm2.onsubmit = function(event) {}:void 🡪 Az automata beöltésére hívódik meg.
* function GetCellValues() {}:void 🡪 Begyűjti a táblázat minden cellájának az elemét és egy map objektumba rendezi. Ha ezzel végzett, akkor meghívja a gráf kirajzolásának függvényét és átadja neki az állapotok számát.
* function finalMapGetter(){}:Map - 🡪 Egy Getter függvény, amely hívásra visszaadja azt a map objektumot, ami tárolja konfigurációs adatokat.
* function hidediv(){} 🡪 Egy Dom objektumnak a láthatóságát állítja. Jelen esetben csak nem láthatóról láthatóra tudja tenni, ezáltal a felhasználónak egy újabb modul megjelenik a főoldalon, ami pedig az állapotdiagram kirajzolás.

### graph.js

### Adattagjai:

* canvas: Adattag, amely értékül kapja a főoldal Dom elemét. Sokszor kerül felhasználásra, ezért van rá szükség.
* contactsPos: Egy globális adattag map, amely tárolja az állapotokat, illetve azok lokalizációs koordinátáit.

### Metódusai:

* MainDraw(qAmount){}:void 🡪 Ez fő függvénye a fájlnak. Ez hívódik meg elsőként. Paraméterül kapja a turing-gép állapotainak a számát. Ezután ahány állapot van, annyit feltölt ContactsPos mappbe. Az állapotokon kívül még változó lokalizációs pozíciókat töltünk fel, hogy a vásznon el legyenek terítve egyenletesen az állapotok. Minden egyes elem mapbe felöltésekor meghívódik az állapot kirajzoló függvény, amely a koordináták alapján és a ContactsPosba feltöltött értékek alapján kirajzolja az állapotokat.
* contactDescribe(from, to,contacts) {}:void 🡪 Paraméterül megkapja a kiindulási és végződési állapotot, illetve ezek koordinátáit. Feladata, hogy az összekötő elemekre szöveget írjon fel. Az összekötő elemek állapotok között találhatók. Minden összekötő elemnek van kiinduló és végpontja. A betáplált utasítások alapján kerülnek meghatározásra az összekötő nyilak kirajzolása. A megadott utasítások azok a szövegek, amelyek a nyilak felé kerülnek. Minden nyíl ezáltal leellenőrizhető, hogy szabad szemmel vajon jó állapotból jó állapotba halad-e, mert a szabálya látható.
* function drawQs(name, x, y) {}:void 🡪 Paraméterül megkapja az állapot nevét és lokalizációs koordinátáit. Ezután a canvas könyvtár függvényei segítségével a vásznon kirajzolódnak az állapotok. Egy függvény hívás egy darab állapotnak a kirajzolását teszi meg.
* contactTo(from, to) {} 🡪 Végzi el a nyilak kirajzolását a pontok között. A nyilak vonalból és egy fejből állnak, ezek kerülnek megrajzolásra. A nyíl feje mindig az érkezés helyéhez kerül, ezzel mutatva az érkezési pontot.

### turingjs.js

Ebben forrásfájlban kerülnek összesítésre a konfigurációs lépések. Az itt megkapott adatok automata betöltése után kerülnek ide.

### Adattagjai:

* steps :{} – Object adattag. Ebbe kerülnek feltöltve a konfigurációs adatok, amely alapján dolgozik az olvasás a szalagon.

### Metódusai:

* Turing (steps, start,end){}:string - Bár ez az objektum nem itt van implementálva, azonban itt van példányosítva. Az új példány pedig a saját metódusait használja működéséhez. Többek között a szalag kezelő, az olvasófej indítója, megállítója, léptetője illetve a szalag inputnak a metódusai.

### turing.js

Ez egy Turing objektum. Metódusai a turing-gép irányításáért felelősek. Adattagjai a kapott paraméterek szerint változtathatók. Metódusait kifejtem a következő sorokban.

### Adattagjai:

* this.steps = steps – A paraméterül kapott steps az, ami tárolja a lépéseket.
* this.start = start – Paraméterül kapott kezdő állapot száma (általlában 0 vagy 1)
* this.end = end – Paraméterül kapja a végállapot számát.
* this.tape = [] – Tömb, a szalag tartalma található benne.
* this.length = 10 – Az animált szalag hossza, négyzetek jelölik a főoldalon.
* this.speed = 100 – Az olvasó sebessége, ha automatikusan kéne, hogy olvasson.
* this.stop = true – Stop állapotát tárolja.
* this.count = 0 – Léptetést segítő inkrementálható adattag.

### Metódusai:

* Turing.prototype.machine = function ():void {} felel a futtatásért. Kezeli annak az állapotát, hogy fut-e a gép, illetve, hogy hogyan fut. Továbbá ezeket az állapotokat állító Dom objektumokat is kezeli.
* Turing.prototype.reset = function ():void {} meghívásakor vissza állítja a gépet a kezdeti állapotba. Az állapotot megáll-ra teszi és lekezeli a többi Dom objektumot.
* Turing.prototype.info = function ():void {} automatikusan meghívva van. A gép aktuális állapotait írja ki a főoldalra. Monitorozza a gép aktuális állapotait.
* Turing.prototype.field = function (value) {} paraméter értékül egy stringet kap. Szalag animálásban segédkezik, hogy mozgónak tűnjön a szalag működése, kirajzolás után törölni is kell. Ez a függvény a törlést végzi.
* Turing.prototype.machine = function () {} A gép agyának is tekinthetjük. Itt kapja meg az olvasó fej által olvasott értéket. Eldönti, hogy értelmezhető-e, ha igen tovább léptet, ha nem, akkor pedig hibát dob.
* Turing.prototype.write = function (step):void {} Kiírja az olvasott értéket, a következő lépést és ezután tovább léptet.
* Turing.prototype.move = function (step):void {} Léptetésért felelős, jobbra vagy balra léptet a szalagon, attól függ, hogy mit olvas a fej. Két opció van, jobbra vagy balra lehet menni mivel két irányú a szalag. Minden egyes olvasásnál a move adattag értéke határozza meg az olvasás következő irányát.
* Turing.prototype.check = function ():void {} Vizsgálja, hogy mikor ér a gép terminált állapotba. Minden alkalommal vizsgálja ezt. Akkor terminál sikeresen, ha az aktuális állapot megegyezik a végállapottal.
* Turing.prototype.step = function ():void {} Kézi léptetés metódusa. Hívásakor az automata léptetést megállítja, és csak egyenként kézzel lehet lépni a szalagon.

## A Kliens-oldal forrás elemei

### Upload.php

### Adattagjai:

* $\_POST['fileAjax'] - Szuperglobális változó melyet egy AJAX hívás tölt meg tartalommal. Jelen esetben egy HTML formmal.
* $\_POST['fileAjaxTable'] - Szuperglobális változó melyet egy AJAX hívás tölt meg tartalommal. Ebben a formában jelen esetben egy HTML formmal.
* $\_FILES['fileAjax'] - Szuperglobális változó melyet egy AJAX hívás tölt meg tartalommal. Jelen esetben ez is egy http POST metódus, de tömbként kódolva, fájl van kerül átadásra

### Metódusai:

1. Lineáris a programozása a forrásfájlnak. Két feltétel van benne, amelyek vizsgálják van-e adat a szuperglobális változókban. Ha van, például egy fájl és annak tartalma, akkor azt visszaküldik konzolra írással.

# Tesztelés

Manuális tesztelés [5.1 táblázat], mely során az input mezőkbe egyesével, kézzel nem megfelelő adatot vagy üres adatot teszünk bele.

5.1. táblázat:

|  |  |  |
| --- | --- | --- |
| Esemény | Elvárt eredmény | Kapott eredmény |
| Input szalagnak ékezetes betűt adni |  |  |
| Input szalagnak számot adni |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Összegzés

Szakdolgozatomban egy oktatói programot szerettem volna létrehozni, amely segítség lenne a diákoknak és a tanároknak egyaránt. E program tervezése során felélénkítettem, nekem milyen nehézségeim voltak a Turing-gép megértésében. Azt gondolom, hogy minden egyetemen van egy kis eltérés a leadott tananyagban, emiatt én a saját oktatási intézményemben tapasztalt szokásokra alapoztam. Ilyen volt például a táblázatba foglalt automaták konfigurációja, valamint ilyen még az állapotdiagram megrajzolása is. Ezek azt gondolom olyan egyedi tanítási módszerek, amikkel más helyen nem feltétlen találkoznék, emiatt terveztem meg úgy a programot, ahogy az itteni tanárok szokták megtanítani a tananyagot.

# Irodalomjegyzék

1. https://web.cs.elte.hu/~tichlerk/logika/h/tg.pdf
2. <https://github.com/fabian/Turing>
3. https://uploadcare.com/blog/file-upload-ajax/
4. <https://stackoverflow.com/questions/25134998/how-to-give-a-unique-id-for-each-cell-when-adding-custom-columns>