  
**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](#_Toc60261403)

[1. Elméleti háttér 4](#_Toc60261404)

[Alapfogalmak 4](#_Toc60261405)

[2. Felhasználói dokumentáció 7](#_Toc60261406)

[A program által megoldott feladat (nem jó ez cím) 7](#_Toc60261407)

[Célközönség (nem jó ez a cím, nem kell ide) 7](#_Toc60261408)

[Rendszerkövetelmények 7](#_Toc60261409)

[Minimális hardveres környezet 7](#_Toc60261410)

[Optimális hardveres környezet 8](#_Toc60261411)

[Szoftveres követelmények 8](#_Toc60261412)

[Üzembe helyezés 8](#_Toc60261413)

[Hordozhatóság 11](#_Toc60261414)

[A program használata 11](#_Toc60261415)

[Turing-gép létrehozás 11](#_Toc60261416)

[Turing-gép szerkesztő és futtató felület 13](#_Toc60261417)

[Konfigurációs panel 14](#_Toc60261418)

[2.1.1. Gráf diagram 15](#_Toc60261419)

[2.1.2. 15](#_Toc60261420)

[2.1.3. Hibák és egyéb üzenetek 16](#_Toc60261421)

[2.1.4. 16](#_Toc60261422)

[3. Fejlesztői dokumentáció 18](#_Toc60261423)

[Tervezés 18](#_Toc60261424)

[Felhasználói esetek 18](#_Toc60261425)

[A rendszer architektúrája 19](#_Toc60261426)

[Kliens oldal 21](#_Toc60261427)

[Alkalmazott technológiák 21](#_Toc60261428)

[Alkalmazott módszerek 23](#_Toc60261429)

[A Kliens-oldal forrás elemei 24](#_Toc60261430)

[Index.html 24](#_Toc60261431)

[Js.js 27](#_Toc60261432)

[Adattagjai: 27](#_Toc60261433)

[Metódusai: 28](#_Toc60261434)

[graph.js 29](#_Toc60261435)

[Adattagjai: 29](#_Toc60261436)

[Metódusai: 29](#_Toc60261437)

[4. Tesztelés 30](#_Toc60261438)

[Irodalomjegyzék 31](#_Toc60261439)

# Bevezetés

A szakdolgozat célja, hogy mások számára vizuális eszközökkel könnyebben megérthető legyen a Turin gép működése. A szakdolgozatnak két fő részre van, az algoritmus implementálása egy programozási nyelvre és 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

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)

Az sok tananyag tanulása során néha kevés idő jut, például egy  
összetettebb algoritmus megértésére. Ez az oktatói program ebben próbál segítséget nyújtani. A gyakorlaton alapul vett gráfos ábrázolással, illetve szalagbemenettel fogja majd illusztrálni a program az algoritmus működését. Ezzel hatékonyabbá válhat a tanulás, egyebeket mellett, például segíthet leellenőrizni az elkészített házifeladat helyességét vagy az órán a tanárnak nem muszáj feltétlen minden alkalommal rajzolni a táblára helyette csak kivetíti ezt a webes alkalmazást.

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

A program elsősorban informatikai karon, azon belül 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 nyújt a hallgatóknak az egyes előre betáplált példa feladatok  
kipróbálásában, ezek újra szerkesztésében. Továbbá segít az oktatóknak is például a zárthelyi feladatok kitalálásában vagy diákok által megírtak kijavításában.

## Rendszerkövetelmények

### Minimális hardveres környezet

Legegyszerűbben személyi számítógépen érdemes használni a programot. Például a piacon amit kapni egyik legolcsóbb ilyen eszköz a Rasperry Pi 4 modell B miniszámítógépen.

* 1,5 GHz (1 500 MHz)
* 4 GB mermória

Már ez is képes futtatni egy web böngészőt és egy webszervert. Tehát 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

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

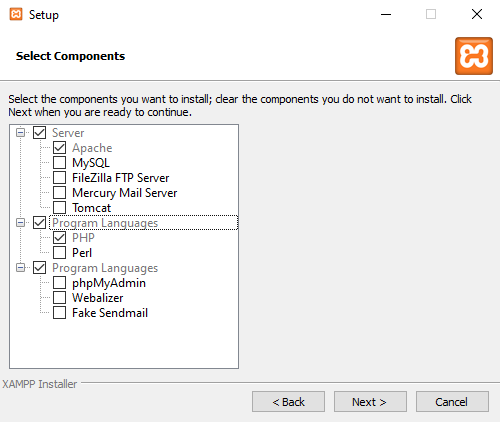
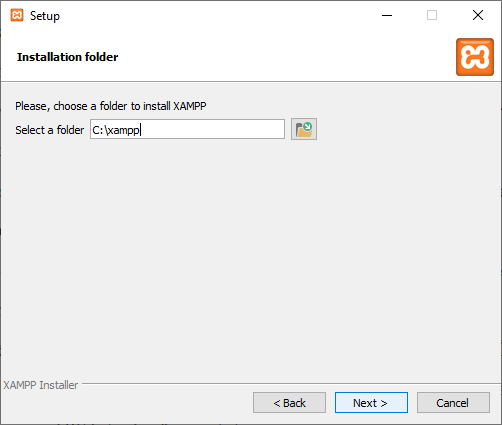
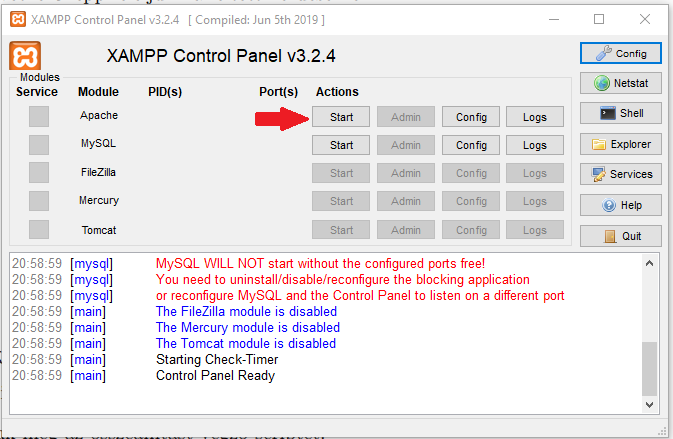
Kliens-oldali futtatáshoz

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 szűkség 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 a csak az Apache webservert 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 telepítés végbement akkor, akkor lehet máris indítani a webszerververt. Ehhez megkell keresni a „xampp-control.exe” indító alkalmazást. Erre kattintva, jelenik meg a control panel ahol az Apache webservernek a start gombjára kell rá kattintani.   
  
* 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 a tenni ami ezen az elérési úton van. C:\xampp\htdocs

Továbbá, ha **szerver számítógépről** szeretnék 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 böngészőbe 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 ellehet érni a programot, feltéve ha nincs jogosultsághoz kötve a szerver látogatása.

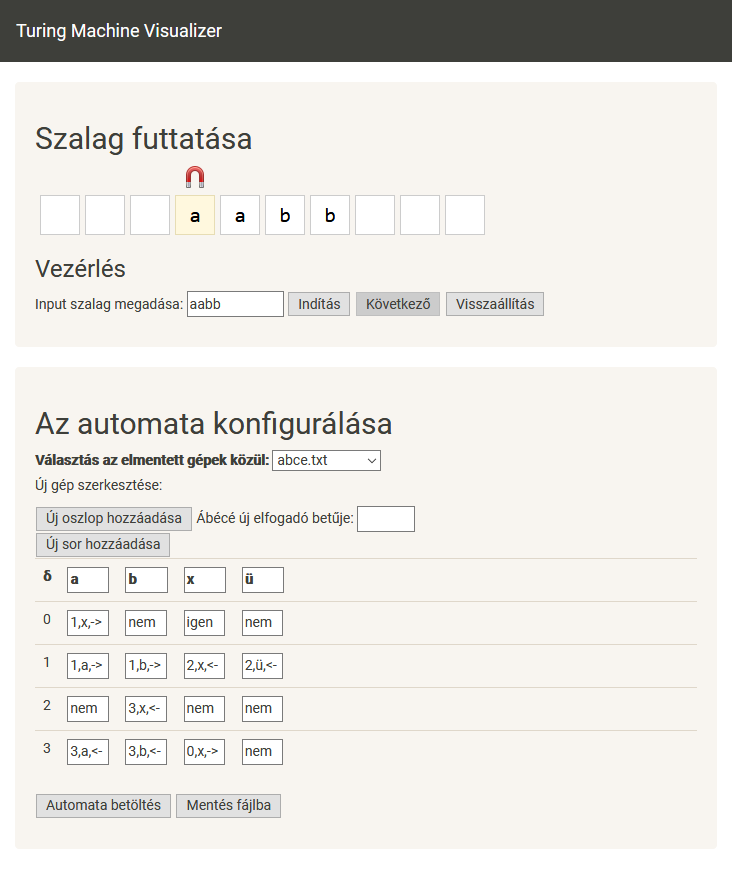
### Hordozhatóság

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

## A program használata

### Turing-gép létrehozás

Személyi számitógépről 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 ez 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 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 oldalján. A két panel sorrendben való kitöltésére nincs szabály megadva, egyetlen lényeges szempont van, hogy minden beviteli mező ki kell töltve legyen.



### 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 nagy betű és szám sem lehet. Az angol ábácé betűt fogadja csal 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*). Lehet megállítani a működésben lévő automatát, lehet léptetni tekerni, újraindítani.



### 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 bármikor újra lehet beinportálni  
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. Új sorokat és oszlopokat lehet hozzáadni. Új oszlopokat csak akkor lehet, ha kötelezően megadunk egy elfogadó betűt amit atáblázat 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 ne legyen duplikáció és növekvő sorrend legyen.

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

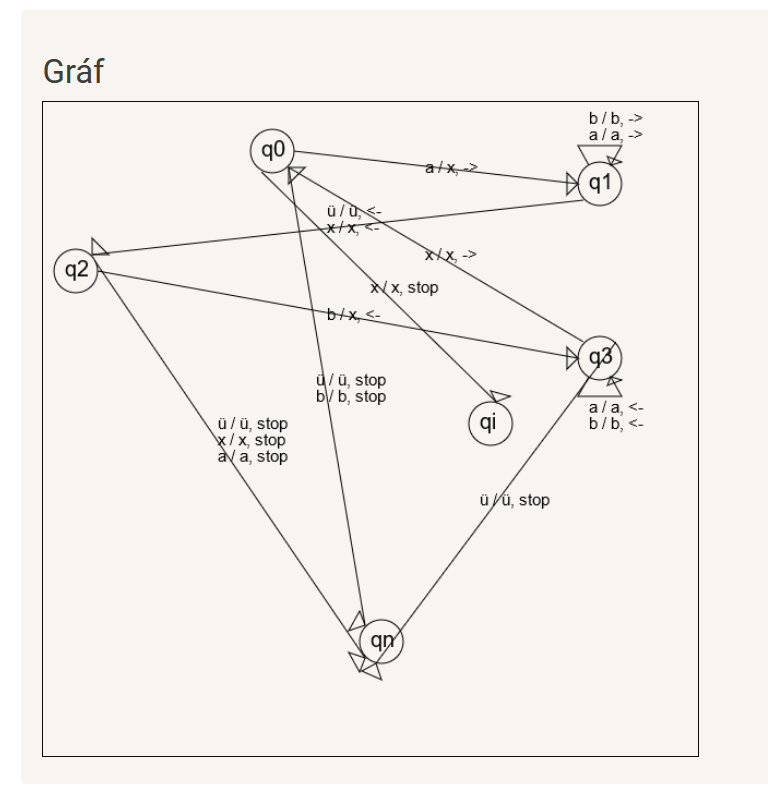
A kész konfigurációt szükség esetén elmenti is lehet egy fájlba (*Mentés fájlba*), amit később újra lehet be lehet importálni.

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



* + 1. Gráf diagram

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



* + 1. Hibák és egyéb üzenetek

Adott esetben megtörténhet, hogy a felhasználó szembesül valamilyen hibajelenséggel. Túlságosan behatóan nem kell ismernie ezeket de nem árt néhány jelenséggel tisztában lennie.

* *„Probléma merült fel az oldal betöltése közben”* Több dolog okozhatja ezt. Leginkább az internet elérésével lehetnek gondok. Ekkor érdemes ellenőrizni az internet kapcsolatot a eszközön amivel próbáljuk futtatni a programot.
* *„Hiba történt a fájlok importálása közben”* Megeshet hogy kész munkánk elveszhetnek, pléldául a vírusírtó által, vagy a program átmozgatása közben. Mindenesetre, a forrásfájlok mellett megtalálható egy mappa ahol az elmentett gépi kunfigurációk találhatók, minden indítás előt 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 szövegbevitelimezők. Érdemes ezeket újból alaposabban átvizsgálni mit rontottunk el.

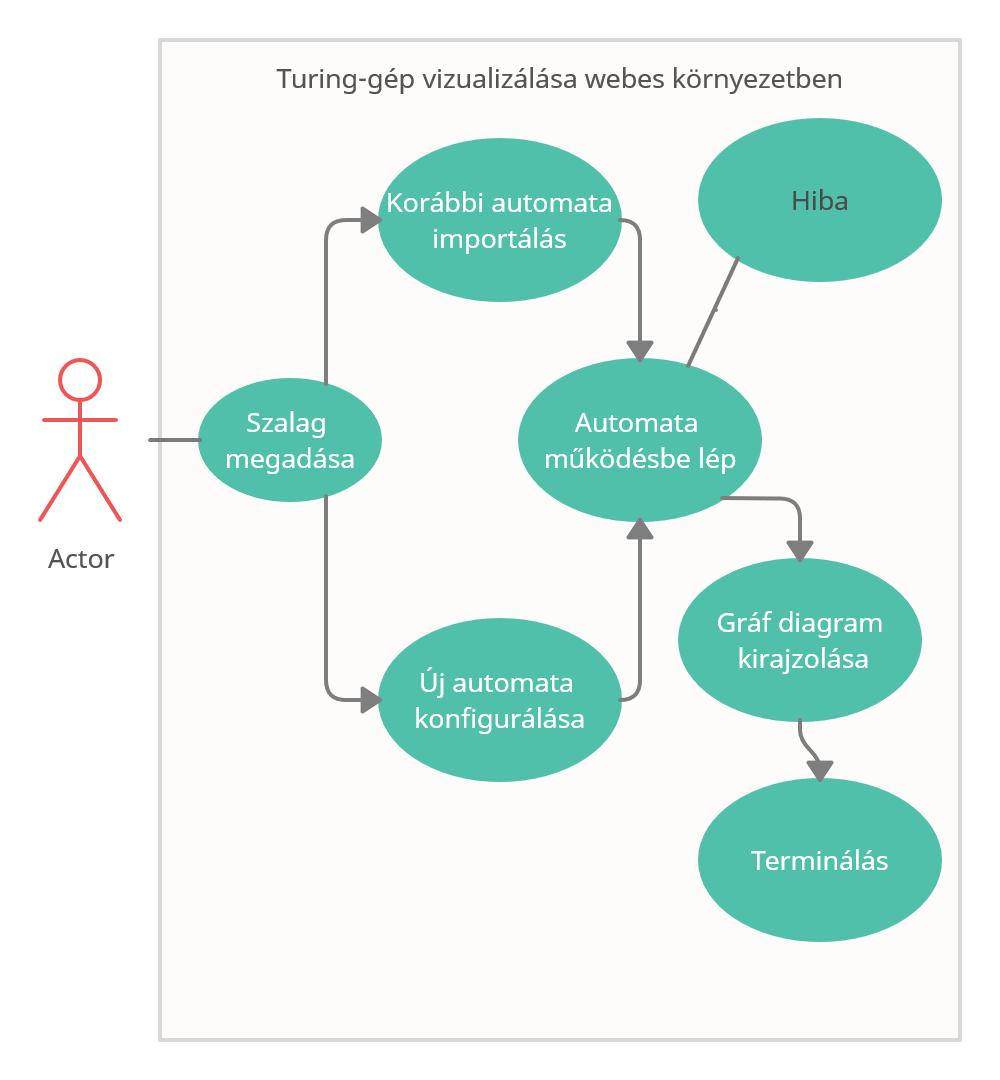
# Fejlesztői dokumentáció

## 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, hogy olyan technológiákat használjak a fejlesztés során, amelyet kötelezően tanítanak az egyetemen, így akár könnyedén továbbfejleszthetővé válik kód.

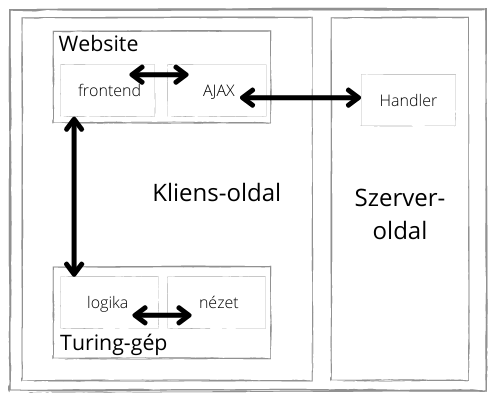
## Felhasználói esetek

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 konfigurlálása ahol megadhatjuk miképeen működjön a gép.



## 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 barátibb megjelenésre tettem a hangsúlyt.



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

Szerver oldalon PHP 7es verzióját használom. Ennek fordítása a hosztoló gépen történek és nem a kliens webbönégszőjében: Emiatt a hosztoló gépen kell lennie PHP 7 forditónak.

#### Klens oldali technológiák amiket használtam

* **HTML:** weboldal leíró nyelv, szabványosított. 5-ös, leújabb 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. A webböngésző fordítja le úgy ahogy 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. JavaScriptet kiegészítésre alkalmas, arra 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 válaszottam ezt a tecnhológiát mert, az alapképzésen a törzs tárgyak között ezt tanították nekünk. A legismertebb szerveroldali szkriptnyelv, 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 model állapotát. Kezeli a felhasználói interakciókat és azokát átadja model számára feldolgozásra

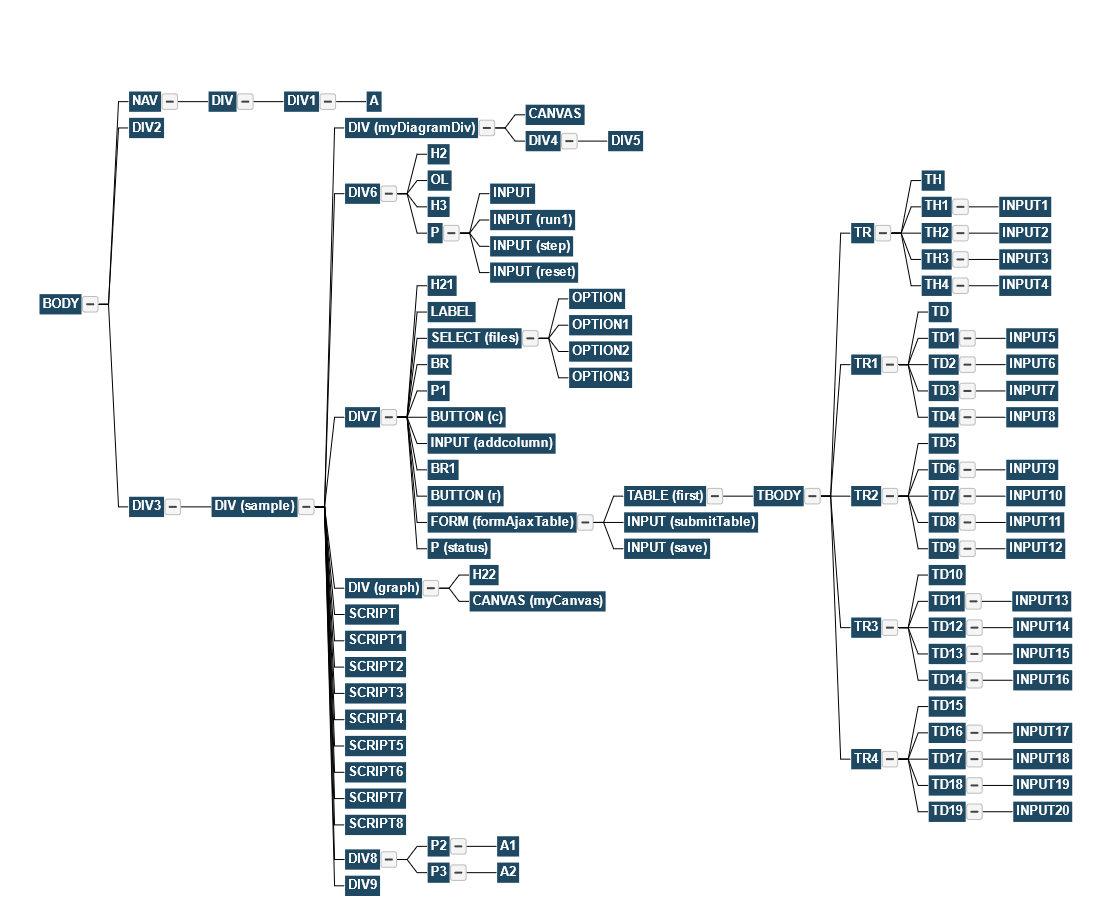
## A Kliens-oldal forrás elemei

* 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 objektumorientáltam programoztam, ezért osztályszerkezeteket nem tudok megjeleníteni, helyette a főoldalam dinamikusan struktúráját tudom bemutatni



#### 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 a beimportáltam a style.css fájt ami mindezt tartalmazza.

A <body> tagben található meg a program 2 fő része. 2 fő részből áll. Az első ahol a szalagot tudjuk betáplálni, itt egy <input text> tag van ami szöveget vár. Csak az angol ábécé betűit, számokat nem fogad el, illetve csak kis betűt. 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, első amikor megy, második mikor áll az olvasó. Ezen kívül az olvasó fej képes automatán olvasni azaz megállni csak ha szalag végén fog megállni. Vagy van a másik mozgása az olvasónak mikor kézzel tudjuk léptetni betűnkként. Ekkor minden gombnyomásra olvas egy betűt és megáll. Tovább, a gombok felett helyezkedik maga szalag. Indítás előtt csak egy üres szalag látszik, indítás után kerül a szalagra a szó amit megadtuk a text input mezőben. indítás után a szalag animálása nem változik, kizárólag az olvasó fej mozog a input szöveg felett és alatt adott esetben a szöveg változik.

Tovább haladva a lapon, megérkezünk konfigurációs panelhez. Itt az első elem egy *drop down list* mely kilistázza mely korábbi turing gépek találhatok a megadott elérési úton. Egyet közölük kiválasztva megkapjuk annak tartalmát és lehet is betölteni az automatába indulásra készen. Ellenben, ha nem szeretnék lementett fájlokat felhasználni akkor készíthetünk új konfigurációt. Ehhez alkottam egy táblázatot mely 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 tartam módosítjuk és nem az oszlopot töröljük. 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 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 állapotdiagrammét. 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.phptől jövő karateráncot íratom ki vele
* *finalMap:* Map ez 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 mely a kapott paraméterek alapján törli a logából a megfelelő oszlopot amit bemeneten kapott
* function reNumber(table, row, i) {}:void Egyik legfontosabb függvény itt. Paraméterül a táblázatot, sort és indexet kap. Ha bármilyen módosítás történek 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 mely oszlop kell adni cellát.
* function addColumnHeader(headerRow) {}:void Táblázat fejlécét szerkeszti az ember, például egy új oszlop esetében akkor ez a függvény helyezi megfelelő helyre a az új oszlop nevet.
* function addColumn(){}: void Gombnyomásra hívódig meg, mikor új oszlopot szeretnék hozzáadni, ez függvény rendezi megfelelő helyre az új oszlopot.
* function addRow(e) {}:void Egy eventet vár. Ha sort szeretnénk a táblázatot 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 a törlését végzi el. Ha utolsó oszlop a táblázatban, akkor azt nem lehet törölni, figyelmeztetést ad
* function getByValue(map, searchValue) {}:string ez egy segédfüggvény, mellyel a egy adott mapben tudok érték alapján keresni, vissza tudja adja 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 mely hívásra visszaadja a 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 mely értékül kapja főoldal Dom elemét. Sokszor kerül felhasználásra, ezért van rá szükség.
* contactsPos egy globális adattag map amely tárolja a az állapotokat, illetve azok lokalizációs koordinátáit.

### Metódusai:

* MainDraw(qAmount){}:void Ez fő függvénye a fájlnak. Ez hívódik meg elsőként. Paraméterül kapja turing gép állapotainak számát. Ezután ahány állapot van annyi, annyit felölt ContactsPos mappbe. Az állapotokon kívül még változó lokalizációs pozíciókat töltünk fel, hogy a vásznon ellegyen terítve egyenletesen az állapotok. Minden egyes elem mapbe felöltésekor meghívódik az állapot kirajzoló függvény, mely a koordináták alapján és a ContactsPosba feltöltött értékek alapján kirajzolja az állapotokat. Állapotok között teszek különbségeket, va
* 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, melyek a nyilak felé kerülnek. Minden nyíl ezáltal leellenőrizhető szabad szemmel jó állapotból jó állapotba megy 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 vászon kirajzolódnak az állapotok. Egy függvény hívás egy darab állapotnak a kirajzolását teszi meg.
* contactTo(from, to) {} végzi el a nyilak kirajzolását pontok között. A nyilak vonalból és egy fejből álnak. 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 mely alapján dolgozik az olvasás a szalagon.

### Metódusai:

* Turing (steps, start,end){}:string - -Bár ez az objektum nem itt van implementálva de 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 metódusai.

### turing.js

Ez egy Turing nevű objektum külön forrás fájlban. Több adattagja és metódusa van. Minde a turing gép irányítására van

### Adattagjai:

* this.steps = steps – a paraméterül kapott steps az ami tárolja a lépéseket
* this.start = start – paraméterül kapott kezdő állapot száma(által
* 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 olvasson
* this.stop = true – stop állapotát tárolja
* this.count = 0 – léptetést segítő inkrementálható adattag

### Metódusai:

* Turing.prototype.machine = function ():void {} felel a futtásért. Kezeli annak az állapotát, hogy fut e a gép illetve hogyan fut. Tovább ezek az állapotokat állító Dom objektumokat is kezeli.
* Turing.prototype.reset = function ():void {} meghívásakor vissza állítja a gépet az kezdeti állapotba. Az állapotot megállra teszi és lekezelő a többi Dom objektumot.
* Turing.prototype.info = function ():void {} automatikusan meghívva van. A gép aktuális állapotait írja ki a főoldalra. Monitorozza 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 függvény a törlést végzi.
* Turing.prototype.machine = function () {} A gép agya ha úgy tetszik. Itt kapja meg az olvasó fej által olvasott az értéket. Eldönti, hogy értelmezhető e, ha igen tovább léptet, ha nem akkor hibát dob
* Turing.prototype.write = function (step):void {} Kiírja az olvasott értéket a következő lépést és tovább léptet.
* Turing.prototype.move = function (step):void {} Léptetésért felelős, jobbra vagy balra léptet a szalagon, attól függ 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 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 formában Jelen esetben egy HTML formmal
* $\_FILES['fileAjax'] - Szuperglobális változó melyet egy AJAX hívás tölt meg tartalommal. Jelen esetben ez egy is egy http POST metódus, de tömb ami jön abban egy fájl van.

### Metódusai:

* Saját maga egy metódus, lineáris a programozása. 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, mely során az input mezőkbe egyesével kézzel nem megfelelelő adatot vagy üres adatot teszünk bele.

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

# Összegzés

Oktatói programot szerettem volna létrehozni, mely segítség lenne a diákoknak és a tanároknak. E program tervezése során felélénkítettem, nekem milyen nehézségeim voltak a Turing gép megértésében. Azt gondolom, hogy minden egyetemen van egy kis eltérés a leadott anyagban, emiatt én a saját oktatási intézményemben tapasztal szokásokra alapoztam. Ilyen volt például a táblázatba foglalt automaták konfigurációja. 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 szeretik megtanítani az anyagot.

# Irodalomjegyzék

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

# Hivatkozások

**Nincsenek források az aktuális dokumentumban.**