IF2124 Teori Bahasa Formal dan Otomata

Laporan Tugas Besar

HTML Checker Menggunakan Bahasa Python

Kelompok : Otax Error

1. Marzuli Suhada M 13522070
2. Bagas Sambega Rosyada 13520071
3. Raden Francisco T. B 13520091

PROGRAM STUDI INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

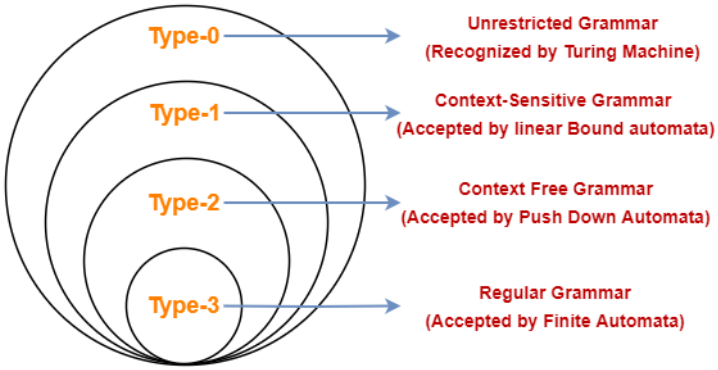
I. Dasar Teori

1.1 Automata Theory

Automata Theory merupakan cabang teori yang menarik dalam ilmu komputer. Cabang ini mengakar pada abad ke-20, ketika para matematikawan mulai mengembangkan mesin - baik secara teoritis maupun nyata - yang meniru beberapa fitur manusia, melakukan perhitungan dengan lebih cepat dan handal. Istilah "automaton" sendiri, yang erat kaitannya dengan "automation", merujuk pada proses otomatis yang melibatkan produksi proses tertentu. Secara sederhana, teori automata berurusan dengan logika komputasi terkait mesin sederhana yang disebut automata. Melalui automata, kita dapat memahami bagaimana mesin menghitung fungsi dan menyelesaikan masalah, dan yang lebih penting, apa artinya sebuah fungsi dapat didefinisikan sebagai dapat dihitung atau sebuah pertanyaan dapat dijelaskan sebagai dapat diputuskan.

Automaton adalah model abstrak dari mesin yang melakukan perhitungan pada suatu input dengan bergerak melalui serangkaian keadaan atau konfigurasi. Pada setiap keadaan perhitungan, fungsi transisi menentukan konfigurasi berikutnya berdasarkan bagian terbatas dari konfigurasi saat ini. Akibatnya, begitu perhitungan mencapai konfigurasi penerimaan, ia menerima input tersebut. Tujuan utama teori automata adalah mengembangkan metode dimana ilmuwan komputer dapat menggambarkan dan menganalisis perilaku dinamis dari sistem diskrit, dimana sinyal diambil sampel secara periodik. Perilaku sistem diskrit ini ditentukan oleh cara sistem tersebut dibangun dari elemen penyimpanan dan kombinasi.

Automaton Family	Type	Grammar (Language)
Finite Automata	3	Regular Grammar
Push Down Automata	2	Context Free Grammar
Linear Bound Automata	1	Context Sensitive Grammar
Turing Machine	0	Unrestricted Grammar



Gambar 1.1 *Chomsky Hierarchy*

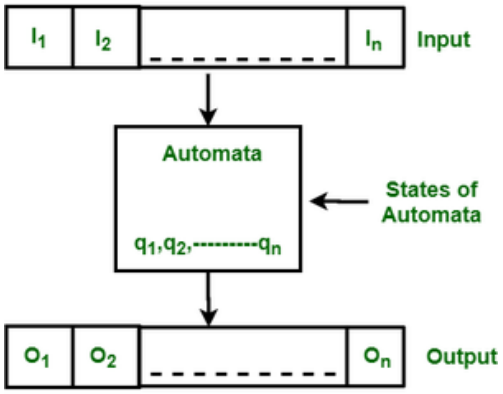
Definisi formal automaton mencakup empat aspek utama:

1. Input, yang berupa kumpulan alfabet (simbol) yang akan diperiksa oleh automaton. Apakah suatu input (kata) diterima oleh automaton ditentukan oleh aturan produksi dan state yang ada. Sebuah kata dianggap ada dalam bahasa jika automaton menerima kata tersebut.
2. States, yang merupakan kumpulan kondisi dari automaton. States dapat berupa finite state, stack, tape, atau bentuk lain sesuai dengan definisi automaton yang bersangkutan.
3. Transition function atau aturan produksi, yang mendefinisikan bagaimana automaton bergerak berdasarkan simbol input dan state tertentu.
4. Acceptance condition, yang menunjukkan kondisi di mana automaton menerima suatu kata (input). Ini dapat berupa final state atau kondisi lain yang telah ditetapkan oleh automaton, seperti berhenti (halt) atau kondisi stack kosong.

1.2 Finite Automata (FA)

Finite Automata (FA) merupakan mesin sederhana untuk mengenali pola dalam Bahasa Regular. Finite Automata didefinisikan dalam bentuk 5 tuple (Q ,Σ , δ,$q0$, F) yaitu :

1. Q : kumpulan state FA yang finite.
2. Σ : kumpulan symbol input yang finite.
3. δ : transition function, yaitu fungsi yang memetakan dari suatu state pada mesin dan suatu input symbol ke state lain pada mesin FA.
4. $q0$: State awal dari mesin, merupakan state pada Q.
5. F : kumpulan state akhir yang diterima oleh FA.



Gambar 1.2 *Features of Finite Automata*

Terdapat dua jenis FA: Deterministic Finite Automata (DFA) dan Nondeterministic Finite Automata (NFA).

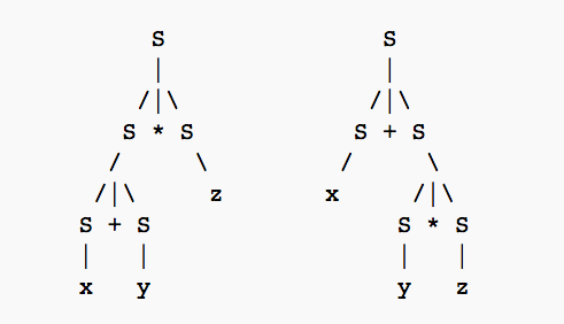
1. DFA terdiri dari lima tupel {Q, Σ, q, F, δ}, di mana setiap simbol input mengarahkan mesin ke satu keadaan. Null move tidak diizinkan. Contoh DFA: konstruksi DFA untuk menerima bahasa string yang berakhir dengan 'a' adalah representasi formal dari DFA.
2. NFA, meskipun mirip dengan DFA, memungkinkan null move dan dapat beralih ke beberapa keadaan untuk input tertentu. Namun, kekuatan keduanya setara, dan setiap NFA dapat diubah menjadi DFA. Penting untuk dicatat bahwa NFA dapat memiliki jalur berbeda untuk satu string input, dan jika salah satu jalur tersebut mencapai keadaan akhir, maka string input diterima oleh NFA.

1.3 Context Free Grammar (CFG)

Context-Free Grammar (CFG) adalah bentuk formal dari tata bahasa yang digunakan untuk mendeskripsikan sintaks atau struktur dari bahasa formal. CFG terdiri dari empat tupel: (V, T, P, S).

1. V: Kumpulan variabel atau simbol non-terminal.
2. T: Sebuah himpunan simbol terminal.
3. P: Aturan produksi yang terdiri dari simbol terminal dan non-terminal.
4. S: Simbol awal.

Sebuah CFG dikatakan sebagai CFG jika setiap produksinya memiliki bentuk: G -> $(V \cup T)^*$, dimana $G \in V$ dan sisi kiri dari G hanya dapat menjadi variabel, bukan terminal. Namun, sisi kanan dapat berupa kombinasi variabel dan terminal.



Gambar 1.3 Contoh CFG

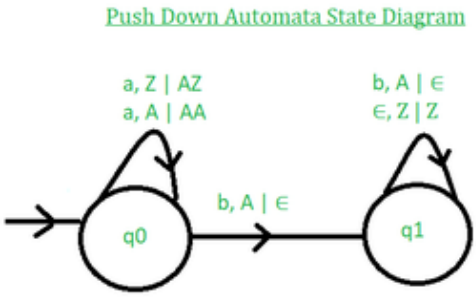
Dalam bidang ilmu komputer, CFG sering digunakan, terutama dalam teori bahasa formal, pengembangan kompilator, dan pemrosesan bahasa alami. CFG juga digunakan untuk menjelaskan sintaks bahasa pemrograman dan bahasa formal lainnya. Namun, ada beberapa keterbatasan CFG:

1. Kurang ekspresif, tidak dapat menggambarkan bahasa Inggris atau bahasa pemrograman.
2. Dapat ambigu, menghasilkan beberapa pohon parse dari input yang sama.
3. Beberapa CFG dapat kurang efisien karena kompleksitas waktu eksponensial.
4. Sistem pelaporan kesalahan kurang tepat dan tidak memberikan informasi detail.

1.4 PushDown Automata (PDA)

Pushdown Automata (PDA) merupakan *Finite Automata* yang dilengkapi dengan memori tambahan berupa tumpukan (stack), yang membantu PDA dalam mengenali *Context Free Grammar.* PDA dapat dijelaskan sebagai berikut:

1. Q adalah himpunan keadaan.
2. $\sum$ adalah himpunan simbol input.
3. Γ adalah himpunan simbol pushdown (dapat ditambahkan dan dihapus dari stack).
4. q0 adalah keadaan awal.
5. Z adalah simbol pushdown awal (yang awalnya ada di dalam stack).
6. F adalah himpunan keadaan akhir.
7. δ adalah fungsi transisi yang memetakan Q x $\{\sum \cup \in\}$ x Γ ke Q x Γ^*. Pada setiap keadaan, PDA akan membaca simbol input dan simbol stack (pada puncak stack) untuk berpindah ke keadaan baru dan mengganti simbol pada stack.



Gambar 1.4 Contoh Diagram PDA

Instantaneous Description (ID) adalah notasi informal tentang bagaimana PDA "menghitung" string input dan membuat keputusan apakah string tersebut diterima atau ditolak. ID adalah triple (q, w, α), dengan q sebagai keadaan saat ini, w sebagai sisa input, dan α sebagai isi stack, dengan puncaknya di sebelah kiri.

Notasi turnstile (⊢) digunakan untuk merepresentasikan satu langkah, sedangkan ⊢* mewakili serangkaian langkah. Sebagai contoh, (p, b, T) ⊢ (q, w, α) berarti dalam melakukan transisi dari keadaan p ke keadaan q, simbol input 'b' dikonsumsi, dan puncak stack 'T' digantikan oleh string baru 'α'.

PDA bersifat deterministik jika terdapat satu langkah dari suatu keadaan untuk simbol input dan simbol stack tertentu. Sebaliknya, PDA non-deterministik dapat memiliki lebih dari satu langkah dari suatu keadaan untuk simbol input dan simbol stack. Meskipun konversi dari PDA non-deterministik ke PDA deterministik tidak selalu mungkin. Esensi dari PDA non-deterministik lebih besar dibandingkan PDA deterministik, dan ada opsi penerimaan melalui stack kosong atau melalui keadaan akhir yang dapat saling dikonversi.

Pengaplikasian PDA pada HTML dapat dijelaskan sebagai berikut:

1. **Definisi Bahasa:** Tentukan bahasa yang ingin diakui oleh PDA. Dalam konteks HTML, bahasa ini dapat berisi aturan-aturan terkait struktur dan sintaks HTML, seperti aturan-aturan untuk tag pembuka dan penutup, hierarki elemen, dll. Pada Tugas Besar ini kami menggunakan bahasa python.
2. **Pembuatan PDA:** Konstruksi PDA yang sesuai dengan bahasa yang telah digunakan. PDA harus memiliki *stack* untuk menyimpan informasi tentang elemen-elemen HTML yang sedang diproses.

3. **Transisi PDA:** Tentukan aturan-aturan transisi untuk PDA. Setiap transisi harus memperhitungkan perubahan status *stack* sesuai dengan elemen-elemen HTML yang diketahui.
4. **Proses Parsing:** PDA digunakan untuk memproses string HTML. Saat PDA membaca karakter HTML satu per satu, ia memutuskan transisi-transisi yang sesuai berdasarkan aturan-aturan bahasa. PDA akan melakukan push dan pop pada *stack* sesuai dengan tag-tag HTML yang diketahui.
5. ***Accepted* atau *Rejected*:** Jika PDA dapat mengakhiri proses parsing dan mencapai kondisi akhir yang sesuai, maka string HTML diterima. Sebaliknya, jika PDA tidak dapat menyelesaikan parsing atau mencapai kondisi kesalahan, maka string HTML ditolak.

II. Hasil PDA

Berikut hasil diagram PDA dengan rincian bahasa seperti berikut:

1. **State:** Q, Q1, Q2, Q3, Q4, …, Q33
2. **Input alphabet:** semua karakter yang dimasukkan oleh pengguna
3. **Stack alphabet:** A, B, C, D, K,L, M, P, Q, R, W, X, Y, <, >, /, “, a, b, c, d, e, f, g, h, i, k, l, n, o, p, r, s, t, u, v, x, y
4. **Start State:** Q
5. **Start Stack Symbol:** Z
6. **Accepting State:** F = {Q99}
7. Keterangan tambahan:
8. **Sigma (Σ)** adalah semua input yang mungkin pada input alphabet

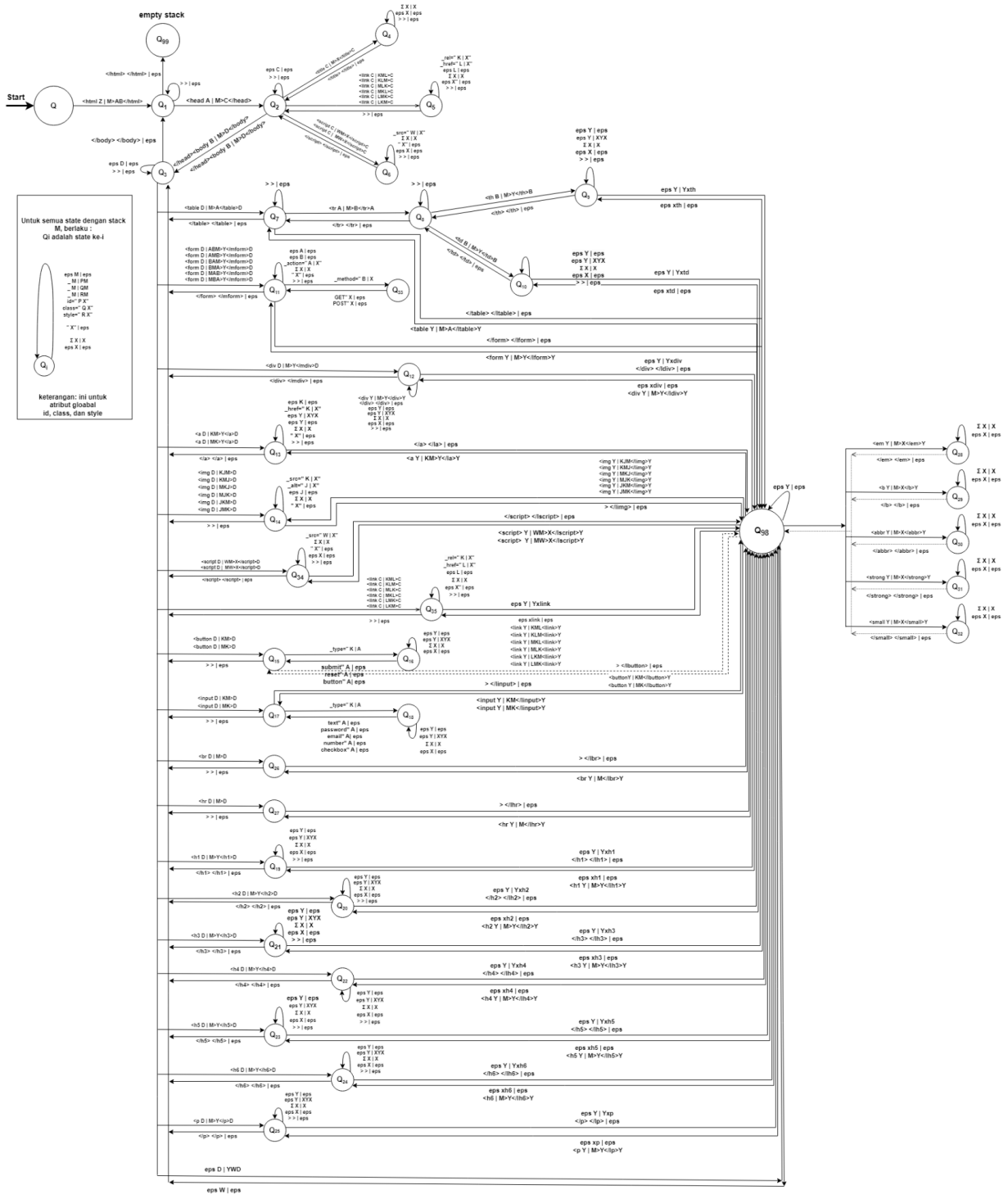
Dalam file txt hasil PDA, sigma(Σ) direpresentasikan sebagai persen(%).

Penjelasan semua state:

- **Q:** start state
- **Q1:** state untuk posisi dalam tag html
- **Q2:** state untuk posisi dalam tag head
- **Q3:** state untuk posisi dalam tag body
- **Q4:** state untuk posisi dalam tag title
- **Q5:** state untuk posisi dalam tag link untuk head
- **Q6:** state untuk posisi dalam tag script untuk head
- **Q7**: state untuk posisi dalam tag table
- **Q8:** state untuk posisi dalam tag tr
- **Q9:** state untuk posisi dalam tag th
- **Q10:** state untuk posisi dalam tag td
- **Q11:** state untuk posisi dalam tag form

- **Q12:** state untuk posisi dalam tag div
- **Q13:** state untuk posisi dalam tag a
- **Q14:** state untuk posisi dalam tag img
- **Q15:** state untuk posisi dalam tag button
- **Q16:** state untuk atribut type pada button
- **Q17:** state untuk posisi dalam tag input
- **Q18:** state untuk atribut type pada input
- **Q19:** state untuk posisi dalam tag h1
- **Q20:** state untuk posisi dalam tag h2
- **Q21:** state untuk posisi dalam tag h3
- **Q22:** state untuk posisi dalam tag h4
- **Q23:** state untuk posisi dalam tag h5
- **Q24:** state untuk posisi dalam tag h6
- **Q25:** state untuk posisi dalam tag p
- **Q26:** state untuk posisi dalam tag br
- **Q27:** state untuk posisi dalam tag hr
- **Q28:** state untuk posisi dalam tag em
- **Q29:** state untuk posisi dalam tag b
- **Q30:** state untuk posisi dalam tag abbr
- **Q31:** state untuk posisi dalam tag strong
- **Q32:** state untuk posisi dalam tag small
- **Q33:** state untuk atribut method pada form
- **Q34:** state untuk posisi dalam tag script untuk head
- **Q35:** state untuk posisi dalam tag link untuk head
- **Q98:** state penyambung semua tag agar dapat melakukan nesting
- **Q99:** accepting state dengan cara empty stack.

Berikut diagram PDA program:



Gambar 2.1 Diagram PDA

III. Implementasi

3.1 Struktur Data

Program yang kami buat terdiri dari dua file utama yaitu main.py dan pda.txt. main.py berfungsi untuk mem-parse file html menjadi string-string yang digabung dalam sebuah list. main.py juga berfungsi untuk mengecek apakah urutan tag sudah sesuai atau belum dan apakah sudah sesuai dengan ketentuan. setelah file input di-parse, dilakukan pengecekan menggunakan file pda.txt lalu di cek kebenarannya dengan menggunakan algoritma PDA (*PushDown Automata*). Algoritma tersebut akan mengecek apakah tag-tag pada file html yang dimasukan sesuai dengan bahasa yang kita buat di pda.txt. Jika bahasa tersebut cocok, maka program akan menampilkan "Accepted". Sedangkan jika ada format seperti urutan yang terbalik atau peletakkan comment yang tidak sesuai pada file html, maka program akan menampilkan "Syntax Error".

3.2 Fungsi dan Prosedur

Fungsi yang digunakan dalam program kami adalah sebagai berikut.

3.2.1 ParseHTML(file)

Mengubah file HTML menjadi string dan juga menghilangkan notasi *new line* dari HTML. Fungsi ini bertujuan untuk melakukan parsing terhadap isi HTML dari sebuah file dan menyimpannya dalam bentuk array. Fungsi parseHTMLfile menerima satu parameter, yaitu filename, yang merupakan nama file HTML yang akan diproses. Fungsi ini membaca isi file, baris per baris, dan melakukan berbagai manipulasi string untuk membersihkan dan menyusun sintaks HTML.

Pertama-tama, fungsi membuka file dan melakukan iterasi melalui setiap baris dalam file. Selanjutnya, untuk setiap karakter dalam baris, fungsi menghilangkan spasi awal dan mengonstruksi string syntax yang merepresentasikan elemen HTML pada baris tersebut. Setelah itu, fungsi membersihkan spasi yang berlebihan dengan mengganti dua spasi berturut-turut dengan satu spasi melalui loop while.

Selanjutnya, terdapat langkah-langkah untuk menghilangkan spasi yang tidak seharusnya pada tag HTML, seperti spasi sebelum atau sesudah tanda kurung sudut (">" atau "<"). Ini dilakukan dengan loop while yang terus berjalan sampai tidak ada lagi pola yang cocok ditemukan.

Terakhir, jika string syntax setelah semua manipulasi tidak kosong dan tidak hanya terdiri dari spasi atau karakter kontrol, maka string tersebut ditambahkan ke dalam array contentofFile setelah dihapus spasi di awal dan

akhirnya. Hasil akhirnya adalah array yang berisi elemen-elemen HTML yang sudah diproses dan dioptimalkan dari file yang diberikan.

3.2.2 deleteComment(file)

Berfungsi untuk menghapus comment yang berada di luar tag dari file HTML. Fungsi ini bertujuan untuk menghapus komentar dari kode HTML. Fungsi ini menggunakan beberapa variabel pengontrol, seperti inTag, commentInsideTag, dan comment, untuk melacak apakah sedang berada di dalam tag HTML, di dalam komentar yang berada di dalam tag, atau sedang berada di dalam komentar di luar tag.

Selama iterasi melalui setiap karakter dalam string HTML, fungsi melakukan pengecekan kondisi untuk menentukan apakah karakter tersebut termasuk dalam tag HTML atau komentar. Jika sedang di luar tag dan menemui awalan komentar (< !), variabel comment diaktifkan, dan jika menemui akhiran komentar (>-), variabel comment dinonaktifkan. Selama dalam komentar, karakter tidak ditambahkan ke dalam string output.

Selanjutnya, jika karakter bukan bagian dari komentar, dan bukan karakter di dalam tag, maka karakter tersebut ditambahkan ke dalam string output. Proses ini berlanjut hingga akhir iterasi melalui string HTML. Hasilnya adalah sebuah string yang merupakan kode HTML tanpa komentar.

3.2.2 removeStrings(file)

Fungsi removeStrings digunakan untuk menghapus string yang terdapat dalam sebuah teks HTML. Implementasinya melibatkan penggunaan variabel kontrol, seperti inside_quotes, yang menandakan apakah saat ini berada di dalam tanda kutip ganda atau tidak. Selama iterasi melalui setiap karakter dalam string HTML, fungsi melakukan pengecekan kondisi. Jika karakter yang ditemui adalah tanda kutip ganda ("), maka variabel inside_quotes akan dibalik (toggle), dan karakter tersebut tetap ditambahkan ke dalam hasil akhir. Jika tidak berada di dalam tanda kutip, karakter akan ditambahkan ke dalam hasil.

Dengan menggunakan variabel kontrol ini, fungsi dapat menyaring karakter-karakter yang berada di dalam tanda kutip ganda, sehingga string yang berada di dalam tanda kutip akan dihilangkan dari hasil akhir. Hasil akhirnya adalah sebuah string HTML yang tidak lagi mengandung teks yang berada di dalam tanda kutip ganda.

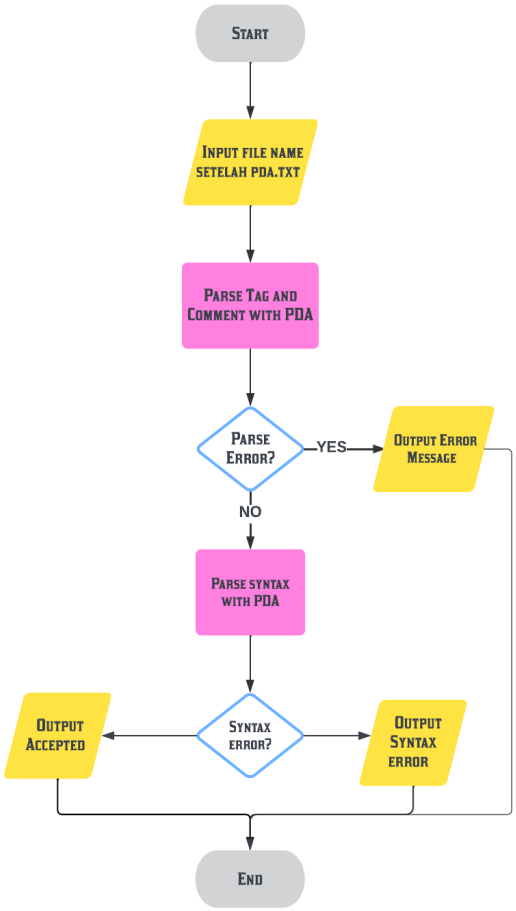
3.3 Antarmuka

Dalam proyek ini, kami menggunakan Antarmuka yang berbasis CLI (*Command Line Interface*). Program ini dapat dioperasikan melalui terminal atau command prompt dengan mengetikkan python main.py pda.txt “file.html”. Penting untuk memastikan bahwa file yang ingin diperiksa berada dalam satu direktori dengan program utama. Setelahnya, hasilnya akan ditampilkan, memberitahu apakah terdapat *syntax error* pada program atau program *accepted*.

```
PS D:\Python\Tubes-IF2124-TBFO\src> py main.py main.txt test1.html
Accepted
```

Gambar 3.3 Tampilan program pada cmd

3.4 Flow Program



Gambar 3.2 Flowchart Program

IV. Pengujian / Program Testing

4.1 Uji 1 (*test1.html*)

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
</html>
```

```
PS D:\Python\Tubes-IF2124-TBFO\src> py main.py main.txt test1.html
Accepted
```

Gambar 4.1 Output pengecekan test1.html

Dalam uji tersebut, file html yang digunakan sudah sesuai dengan ketentuan yang berlaku sehingga pesan yang dihasilkan adalah diterima.

4.2 Uji 2 (*easy.html*)

```
<html>
</head>
</head>
<body>
</body>
</html>
```

```
PS D:\Python\Tubes-IF2124-TBFO\src> py main.py main.txt easy.html
Syntax error
```

Gambar 4.2 Output pengecekan easy.html

Terdapat kesalahan pada tag pembuka </head> yang seharusnya tidak menggunakan tanda ‘/’.

4.3 Uji 3

```
<html  >
    <head>
        <!-- Ini adalah komentar -->
        <title>
            Website Contoh
        </title> <!-- Ini juga komentar -->

        <!-- Ini juga komentar -->  <link href="pda.txt" rel="stylesheet">
        <link  href="ciko.txt"  rel="stylesheet"> <link href="test1.html"
src="test1.html" rel="stylesheet">
    </head>
    <!--
     Longer
    comment  -->
<body>
<p>Test aja ini mah --></p>
<!--
comment lagi -->
</body>
</html>
```

```
PS D:\Python\Tubes-IF2124-TBFO\src> py main.py main.txt tc1.html
Accepted
```

Gambar 4.3 Output pengecekan tc1.html

Pada pengujian ketiga ini, file tc.1 html mengeluarkan output "*accepted*" karena tag-tag dan komentar sudah sesuai dengan ketentuan yang berlaku.

V. Link Repository

Berikut link repository kami :
https://github.com/zultopia/Tubes-IF2124-TBFO

VI. Link Diagram State

Berikut link diagram state kami :
https://app.diagrams.net/#G1CbVagQ2CuBhSvng2pcsUx9kJYFlHLXqB

VII. Pembagian Tugas

No.	NIM	Nama	Tugas
1.	13522070	Marzuli Suhada M	Grammar

2.	13522071	Bagas Sambega Rosyada	Parser
3.	13522085	Raden Francisco Trianto B.	Grammar

VIII. Daftar Referensi

https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html

https://www.geeksforgeeks.org/introduction-of-finite-automata/

https://www.geeksforgeeks.org/introduction-of-pushdown-automata/

https://www.geeksforgeeks.org/what-is-context-free-grammar/