# MEMBUAT RANCANGAN MESIN HIPOTETIK UNTUK MEMECAHKAN PERSOALAN-PERSOALAN YANG BERKAITAN DENGAN DERETAN SIMBOL NON-DETERMINISTIC FINITE AUTOMATA (NFA)



**Nama Dosen Pembimbing:**Badriah Nursakinah S.T., M.Kom

**Nama Anggota Kelompok 4:**   
Aisyah Mawar Kusuma Salsabilla 201011400704  
Andri Firman Saputra 201011402125  
Fariz Septiawan 201011401491  
Muhammad Nur Ikmalul Ilmi 201011401024  
Roni Sefia 201011401617  
Yogi Rizky Pangestu 201011400705

**TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS PAMULANG**

Jl. Puspitek, Buaran, Kec. Pamulang, Telp (021)7412566, Fax. (021)7412566  
Tangerang Selatan – Banten 15310

DAFTAR ISI

[DAFTAR ISI i](#_Toc120308723)

[DAFTAR GAMBAR iii](#_Toc120308724)

[BAB I PENDAHULUAN 1](#_Toc120308725)

[1.1. Latar Belakang 1](#_Toc120308726)

[1.2. Rumusan Masalah 2](#_Toc120308727)

[1.3. Tujuan Penelitian 2](#_Toc120308728)

[1.4. Manfaat Penelitian 2](#_Toc120308729)

[BAB II PEMBAHASAN 3](#_Toc120308730)

[2.1. Pengertian Automata 3](#_Toc120308733)

[2.2. Pengertian Finite State Automata (FSA) 3](#_Toc120308734)

[2.2.1. Karakteristik Finite State Automata 4](#_Toc120308739)

[2.2.2. Cara Kerja Finite State Automata 4](#_Toc120308740)

[2.2.3. Jenis-jenis Finite State Automata 4](#_Toc120308741)

[2.3. Pengertian Mesin Turing 6](#_Toc120308742)

[2.4. Pengertian ASCII 7](#_Toc120308743)

[BAB III METODE PENELITIAN 8](#_Toc120308744)

[3.1. Waktu Penelitian 8](#_Toc120308746)

[3.2. Alat dan Bahan 8](#_Toc120308747)

[3.2.1. Perangkat Keras (Hardware) 8](#_Toc120308748)

[3.2.2. Perangkat Lunak (Software) 8](#_Toc120308749)

[3.3. Metode Penelitian 9](#_Toc120308755)

[3.4. Metode Pengumpulan Data dan Analisis 9](#_Toc120308756)

[3.5. Metode Pengembangan Perangkat Lunak 9](#_Toc120308757)

[3.6. Persyaratan Sistem 11](#_Toc120308766)

[3.7. Implementasi Sistem 11](#_Toc120308767)

[BAB IV PENUTUP 14](#_Toc120308768)

[4.1. Kesimpulan 14](#_Toc120308773)

[4.2. Saran 14](#_Toc120308774)

[DAFTAR PUSTAKA 15](#_Toc120308775)

[LAMPIRAN 16](#_Toc120308776)

[Source Code 16](#_Toc120308777)

# DAFTAR GAMBAR

[Gambar 1 Contoh Gambar DFA 5](#_Toc120308778)

[Gambar 2 Contoh Gambar NFA 6](#_Toc120308779)

[Gambar 3 Table ASCII 7](#_Toc120308780)

[Gambar 4 Tampilan saat aplikasi dibuka 11](#_Toc120308781)

[Gambar 5 Tampilan saat pengguna diminta memasukkan simbol 12](#_Toc120308782)

[Gambar 6 Contoh memasukkan simbol ababb 12](#_Toc120308783)

[Gambar 7 Tampilan pengguna keluar aplikasi 13](#_Toc120308784)

# BAB I PENDAHULUAN

## Latar Belakang

Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*textprocessor*). Bahasa formal adalah kumpulan kalimat. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama. Sebuah bahasa formal bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda. Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya. Bahasa manusia bersifat sebaliknya. *Grammar* diciptakan untuk meresmikan kata-kata yang hidup di masyarakat. Dalam pembicaraan selanjutnya ‘bahasa formal’ akan disebut ‘bahasa’ saja. Teori bahasa *automata* dapat dijadikan suatu gagasan mendasar dalam komputasi yang menjadi alat untuk mengenali suatu persoalan atau masalah karena dapat memberikan konsep dan prinsip untuk memahami suatu persoalan yang berkorelasi dengan bidang ilmu komputer. Misalnya permasalahan mengenai deretan simbol.

Munculnya permasalahan mengenai deretan simbol *Non-deterministic Finite Automata* (*NFA*) dapat diselesaikan menggunakan mesin hipotetik. Topik ini dipilih untuk memberikan suatu solusi dengan membuat rancangan mesin hipotetik dan aplikasi deretan simbol *NFA* untuk sarana penyelesaian masalah mengenai persoalan-persoalan yang berkaitan dengan deretan simbol *NFA*.

Berdasarkan latar belakang di atas, maka topik yang diambil pada penelitian ini adalah “Membuat Rancangan Mesin Hipotetik Untuk Memecahkan Persoalan-Persoalan Yang Berkaitan Dengan Deretan Simbol *Non-deterministic Finite Automata*”. Diharapkan dari pembuatan aplikasi mesin hipotetik ini, memberikan solusi dalam memecahkan persoalan-persoalan yang berkaitan dengan deretan simbol *NFA* nantinya.

## Rumusan Masalah

Rumusan masalah dari pengembangan aplikasi ini adalah:

1. Masih sedikit dikembangkannya aplikasi pemecahan masalah yang efektif mengenai pengurutan simbol.
2. Kurangnya materi pembelajaran tentang mesin hipotetik.

## Tujuan Penelitian

Berdasarkan permasalahan topik di atas, tujuan dari penelitian ini adalah:

1. Mempelajari dan mengenal tentang pembuatan aplikasi dan mesin hipotetik untuk memecahkan masalah mengenai deretan simbol *NFA*.
2. Mengetahui lebih dalam lagi mengenai mesin hipotetik.

## Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Bagi penulis untuk menambah wawasan dan pengetahuan dalam merancang sebuah mesin hipotetik dan aplikasi.
2. Bagi pengguna, dari hasil penelitian ini diharapkan dapat membantu memecahkan masalahnya mengenai deretan simbol *NFA*.

# BAB II PEMBAHASAN



## Pengertian Automata

*Automata* adalah mesin abstrak yang dapat mengenali, menerima atau membangkitkan sebuah kalimat dalam bahasa tertentu. *Automata* berkaitan erat dengan teori bahasa formal. Selain itu juga ada beberapa hal yang berkaitan dengan *automata*, yaitu *grammar*. *Grammar* adalah bentuk abstrak yang dapat diterima untuk membangkitkan suatu kalimat *automata* berdasarkan suatu aturan tertentu.

## Pengertian Finite State Automata (FSA)

*Finite State Automata* atau *Finite State Machine* adalah mesin abstrak yang memiliki lima elemen atau *tuple*. Kelima elemen tersebut meliputi *input*, *output*, himpunan *state*, relasi *state*, dan relasi *output*. *FSA* berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata. *FSA* dapat menerima *input* dan mengeluarkan *output* yang memiliki *state* yang berhingga banyaknya. Selain itu, *FSA* memiliki sekumpulan aturan untuk berpindah dari satu *state* ke *state* lain berdasarkan *input* dan fungsi transisi yang diterapkan.

Perlu diketahui bahwa sistem *FSA* hanya dapat mengingat *state* terkini karena tidak memiliki tempat penyimpanan/*memory*. *FSA* pada dasarnya digunakan untuk mengenali pola. Dibutuhkan *string* simbol sebagai *input* dan statusnya berubah sesuai dengan *input* tersebut. Ketika ditemukan simbol *input* yang diinginkan, maka terjadi transisi. Pada saat transisi, *automata* dapat berpindah ke keadaan berikutnya atau tetap dalam keadaan yang sama. *FSA* memiliki dua status, status terima atau status tolak. Ketika *string input* berhasil diproses, dan *automata* mencapai *state* akhir, maka statusnya adalah terima, demikian juga sebaliknya.

*FSA* dapat didefinisikan dengan persamaan berikut:

M = (Q , Σ , δ , S , F)

* Q : himpunan *state*
* Σ : himpunan simbol *input*
* δ : fungsi transisi δ : Q × Σ
* S : *state* awal / *intital state*, S ∈ Q
* F : *state* akhir, F ⊆ Q



### Karakteristik Finite State Automata

*Finite State Automata* memiliki beberapa karakteristik berikut:

* Setiap *Finite Automata* memiliki keadaan dan transisi yang terbatas.
* Transisi dari satu keadaan ke keadaan lainnya dapat bersifat *deterministik* atau *non-deterministik*.
* Setiap *Finite Automata* selalu memiliki keadaan awal.
* *Finite Automata* dapat memiliki lebih dari satu keadaan akhir.
* Jika setelah pemrosesan seluruh *string*, keadaan akhir dicapai, artinya *automata* menerima *string* tersebut.

### Cara Kerja Finite State Automata

*Finite State Automata* bekerja dengan cara mesin membaca memori masukan berupa *tape* yaitu 1 karakter tiap saat (dari kiri ke kanan) menggunakan *head* baca yang dikendalikan oleh kotak kendali *state* berhingga di mana pada mesin terdapat sejumlah *state* berhingga.

*Finite Automata* selalu dalam kondisi yang disebut state awal (*initial state*) pada saat *Finite Automata* mulai membaca *tape*. Perubahan *state* terjadi pada mesin ketika sebuah karakter berikutnya dibaca.

Ketika *head* telah sampai pada akhir *tape* dan kondisi yang ditemui adalah *state* akhir, maka *string* yang terdapat pada *tape* dikatakan diterima *Finite Automata* (*String-string* merupakan milik bahasa bila diterima *Finite Automata* bahasa tersebut).

### Jenis-jenis Finite State Automata

*State Finite Automata* dapat dibagi menjadi dua jenis, yaitu:

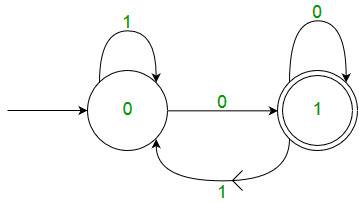
* *Deterministic Finite Automata* (*DFA*)
* *Non-deterministic Finite Automata* (*NFA*)

Kedua *finite automata* di atas mampu mengenali himpunan *reguler* secara presisi. Dengan demikian kedua *finite automata* itu dapat mengenali *string-string* yang ditunjukkan dengan ekspresi *reguler* secara tepat.

*DFA* dapat menuntun *recognizer* (pengenal) lebih cepat dibanding *NFA*. Namun demikian, *DFA* berukuran lebih besar dibanding *NFA* yang ekivalen dengannya. Lebih mudah membangun *NFA* dibanding *DFA* untuk suatu bahasa, namun lebih mudah mengimplementasikan *DFA* dibanding *NFA*.

#### Deterministic Finite Automata (DFA)

Dalam *Deterministic Finite Automata* (*DFA*), untuk karakter input tertentu, mesin hanya dapat menuju satu *state* dan fungsi transisi dipakai pada setiap *state* untuk setiap simbol *input*. Selain itu, pada *DFA*, perpindahan *state null* (atau ε) tidak diperbolehkan, artinya, *DFA* tidak dapat mengubah *state* tanpa karakter *input* sama sekali. *DFA* juga memiliki atas 5 tuple. Sebagai contoh berikut adalah *DFA*   
Σ = (0, 1) yang menerima semua *string* yang diakhiri dengan 0.



Gambar 1 Contoh Gambar DFA

Satu hal penting yang perlu diperhatikan pada jenis *DFA* adalah terdapat banyak kemungkinan pada sebuah pola. Namun secara umum, *DFA* dengan jumlah *state* minimum cenderung lebih baik.

#### Non-deterministic Finite Automata (NFA)

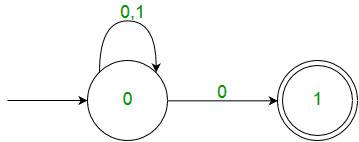
*NFA* hampir sama seperti *DFA*, namun yang membedakan adalah pada beberapa hal berikut:

* Diperbolehkan terjadi perpindahan *null* (atau ε), yang berarti *NFA* dapat berpindah ke *state* berikutnya tanpa membaca simbol *input*.
* *NFA* dapat mengirimkan beberapa *state* untuk *input* tertentu.

Pada dasarnya beberapa hal di atas tidak membuat *NFA* lebih unggul dari *DFA*. Jika kita membandingkan keduanya dalam hal kekuatan, keduanya setara. Karena fitur tambahan di atas, *NFA* memiliki fungsi transisi yang berbeda, selebihnya sama dengan *DFA*. Fungsi transisi pada *NFA* dapat didefinisikan sebagai berikut:

δ : Q x (Σ U ε) → 2 ^ Q.

Berikut adalah contoh dari *NFA*:



Gambar 2 Contoh Gambar NFA

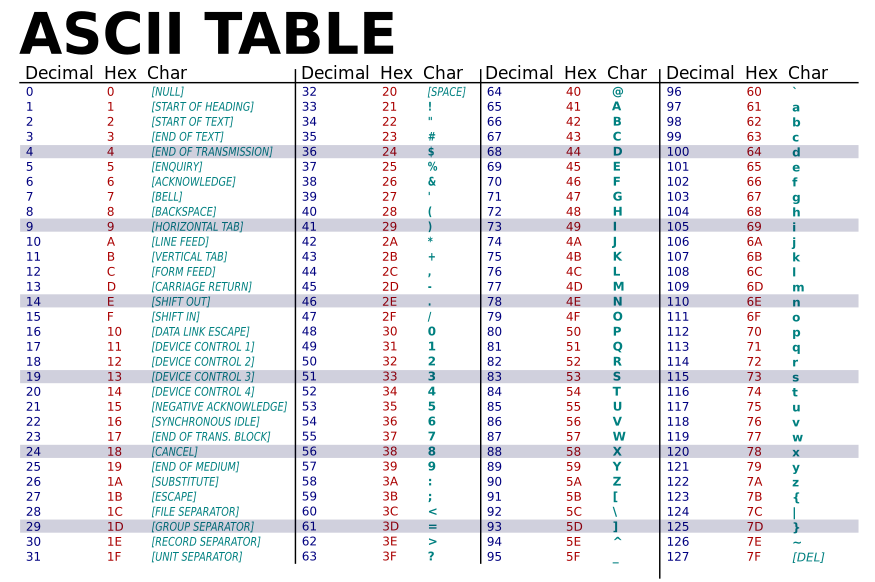
Satu hal penting yang perlu diperhatikan pada *NFA* adalah jika ada jalur untuk *string input* yang mengarah ke *state* akhir, maka *string input* diterima. Misalnya, di *NFA* di atas, ada beberapa jalur untuk *string input* "00". Karena salah satu jalur mengarah ke *state* akhir, "00" diterima oleh *NFA*.

## Pengertian Mesin Turing

Mesin *turing* adalah model komputasi teoretis yang ditemukan oleh *Alan Turing*, berfungsi sebagai *model* ideal untuk melakukan perhitungan matematis. Walaupun *model* ideal ini diperkenalkan sebelum komputer nyata dibangun, *model* ini tetap diterima oleh kalangan ilmu komputer sebagai *model* komputer yang sesuai untuk menentukan apakah suatu fungsi dapat selesaikan oleh komputer atau tidak (menentukan *computable function*). Mesin *turing* terkenal dengan ungkapan “Apapun yang bisa dilakukan oleh Mesin *turing* pasti bisa dilakukan oleh komputer.”

## Pengertian ASCII

*ASCII* singkatan dari (*American Standard Code for Informasi Interchange*). Komputer hanya dapat memahami nomor, maka kode *ASCII* adalah representasi numerik dari karakter seperti ‘a’ atau ‘@’ atau tindakan dari beberapa macam. *ASCII* dikembangkan lama dan sekarang non-cetak karakter yang jarang digunakan untuk tujuan asli mereka. *ASCII* sebenarnya dirancang untuk digunakan dengan *teletypes*. *ASCII* adalah standar pengkodean karakter untuk alat komunikasi. Kode *ASCII* mewakili teks dalam komputer, peralatan telekomunikasi, dan perangkat lainnya.



Gambar 3 Table ASCII

Gambar di atas adalah gambar dari tabel *ASCII*. Pada tabel tersebut terdapat *char* atau karakter yang memiliki kode *decimal* dan kode *hex*. Kode *decimal* itu yang akan digunakan sebagai acuan dari pengurutan simbol yang akan digunakan pada aplikasi.

# BAB III METODE PENELITIAN



## Waktu Penelitian

Waktu penelitian dilakukan pada tanggal 29 Oktober 2022 sampai dengan tanggal 25 November 2022.

## Alat dan Bahan

Dalam pembuatan aplikasi dan mesin *turing* ini, alat dan bahan yang digunakan meliputi perangkat keras, perangkat lunak serta bahan-bahan penunjang lainnya.

### Perangkat Keras (Hardware)

Perangkat keras yang digunakan adalah *laptop* dengan spesifikasi berikut:

1. *Laptop Lenovo Ideapad 320 14IKB-80XK*
2. *Processor Intel(R) Core (TM) i5-7200U CPU @ 2.50 GHz, 2712 Mhz, 2 Core(s), 4 Logical Processor(s)*
3. *RAM 8 GB DDR4 Sodimm 2133MHz dual channel (4GB x 2)*
4. *SSD Verbatim SATA III 240 GB Vi500 2.5”*
5. *WiFi IndiHome 20 Mbps*

### Perangkat Lunak (Software)

Perangkat lunak adalah program yang digunakan untuk menjalankan perangkat keras. Tanpa adanya perangkat lunak ini komponen perangkat keras tidak dapat berfungsi.

Adapun *software* yang digunakan dalam pembuatan permainan ini adalah:

1. Sistem Operasi *Windows 10 Pro*
2. *Microsoft Office Professional Plus 2019*
3. *Dev C++ 5.11*
4. *Draw.io*
5. dan program-program lain yang mendukung pembuatan aplikasi.

## Metode Penelitian

Metode penelitian merupakan suatu cara yang dapat digunakan untuk mencapai tujuan yang diharapkan melalui suatu penelitian dengan teknik-teknik dan alat-alat tertentu. Adapun metode yang digunakan dalam penelitian ini yaitu metode *development research* suatu kegiatan penelitian yang bertujuan dan berusaha mengembangkan atau melengkapi pengetahuan yang sudah ada atau diketahui. Permasalahan manusia dan lingkungan alamnya selalu berkembang yang ke semuanya ini harus memperoleh jawaban yang simbang. (Supardi, 2005:25)

Metode pengurutan simbol yang digunakan pada pembangunan aplikasi adalah pengurutan karakter berbasis karakter kode *ASCII decimal*. Metode pengurutan simbol ini dipilih karena kode *ASCII* adalah kode karakter standar yang digunakan dalam dunia komputer.

## Metode Pengumpulan Data dan Analisis

1. Internet  
   Melakukan pencarian pada website antar universitas yang membahas tentang mesin *turing*, kode *ASCII* dan juga persoalan-persoalan yang berkaitan dengan deretan simbol.
2. Studi Pustaka

Mencari teori dan informasi yang berhubungan dengan topik yang akan dibuat. Pencarian teori dan informasi dicari melalui internet, dan hasil penelitian maupun karya ilmiah.

## Metode Pengembangan Perangkat Lunak

Untuk metode pengembangan perangkat lunak, penulis menggunakan metode *prototyping*. *“Prototype”* adalah implementasi bagian dari produk *software* yang secara *typical* fungsinya dibatasi, realibitas rendah, tampilannya rendah, dan kurang ketegasan. *Prototype* sering dikembangkan secara cepat dalam bahasa tingkat tinggi atau bahasa *prototype* tertentu, tanpa memperhatikan kebenaran dan ketegapan dan sebagainya. (Al Bahra Bin Ladjamudin, 2006:22)

Tahapan-tahapan dalam *Prototyping* adalah sebagai berikut:

1. Pengumpulan Kebutuhan

Pengguna dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

1. Membangun *Prototyping*

Membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pengguna (misalnya dengan membuat *input* dan *output*).

1. Evaluasi *Protoptyping*

Evaluasi ini dilakukan oleh pengguna apakah *prototyping* yang sudah dibangun sudah sesuai dengan keinginan pengguna. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak *prototyping* di revisi dengan mengulangi langkah 1, 2, dan 3.

1. Pengkodean Sistem

Dalam tahap ini *prototyping* yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman *C++*.

1. Menguji Sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, aplikasi harus di coba terlebih dahulu sebelum digunakan. Pengujian ini dilakukan dengan *White Box*, *Black Box*, *Basis Path*, pengujian arsitektur dan lain-lain.

1. Evaluasi Sistem

Pengguna mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika ya, langkah 7 dilakukan; jika tidak, ulangi langkah 4 dan 5.

1. Menggunakan Sistem

Perangkat lunak yang telah diuji dan diterima pengguna siap untuk digunakan.



## Persyaratan Sistem

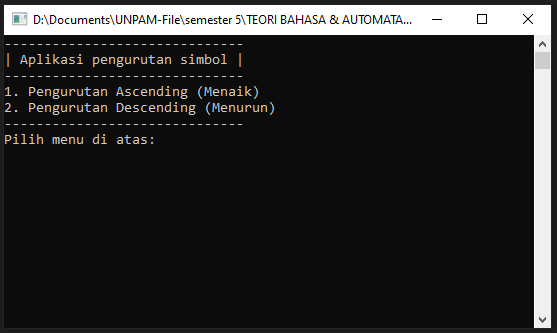
Aplikasi dirancang untuk berjalan pada *platform windows*, guna meningkatkan performansi. Berikut adalah spesifikasi minimal rekomendasi dari pengembang harus dimiliki:

1. *RAM* minimal *1GB*
2. *CPU Cores 2* berkecepatan *1 GHz* atau lebih
3. *OS Windows 7* atau lebih
4. Penyimpanan *internal 1GB*

## Implementasi Sistem

Berikut adalah implementasi sistem dari penggunaan aplikasi

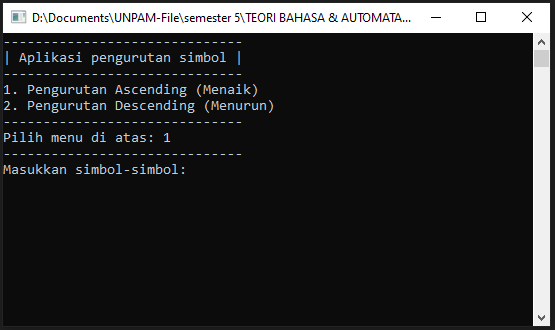
1. Membuka aplikasi



Gambar 4 Tampilan saat aplikasi dibuka

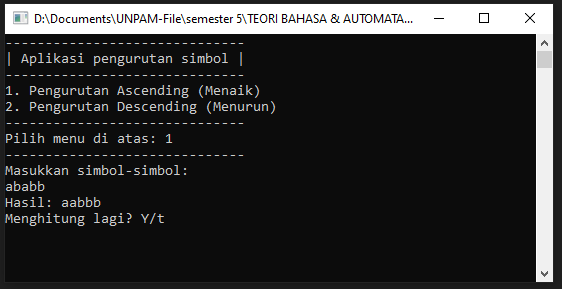
Saat aplikasi terbuka pengguna akan diminta untuk memilih menu. Jika pengguna memasukkan angka 1 maka aplikasi akan melakukan pengurutan string menaik (*ascending*) akan tetapi jika pengguna memasukkan angka 2 maka aplikasi akan melakukan pengurutan menurun (*descending*).

1. Jika pengguna memasukkan angka 1



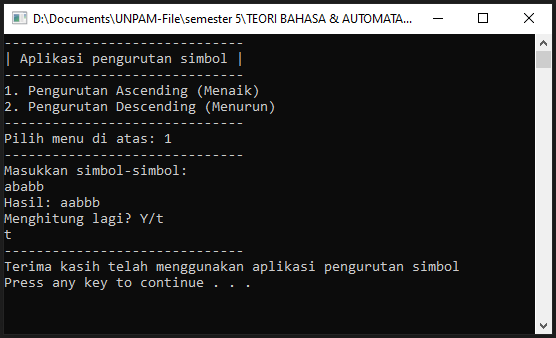
Gambar 5 Tampilan saat pengguna diminta memasukkan simbol

Setelah pengguna memilih menu, pengguna akan diminta memasukkan simbol-simbol yang ingin diurutkan.



Gambar 6 Contoh memasukkan simbol ababb

Dari gambar di atas, sebagai contoh pengguna jika memasukkan simbol “ababb” menjadi simbol “aabbb”. Setelah itu, pengguna diminta memasukkan huruf y jika ingin menghitung lagi dan memasukkan huruf t jika pengguna ingin keluar aplikasi.



Gambar 7 Tampilan pengguna keluar aplikasi

Berikut adalah contoh dari implementasi sistem aplikasi pengurutan simbol untuk *NFA*.

# BAB IV PENUTUP



## Kesimpulan

*Automata* adalah mesin abstrak yang dapat mengenali, menerima atau membangkitkan sebuah kalimat dalam bahasa tertentu. *Automata* berkaitan erat dengan teori bahasa formal. *Finite State Automata* atau *Finite State Machine* adalah mesin abstrak yang memiliki lima elemen atau *tuple*. Kelima elemen tersebut meliputi *input*, *output*, himpunan *state*, relasi *state*, dan relasi *output*. Diperbolehkan terjadi perpindahan *null* (atau ε), yang berarti *NFA* dapat berpindah ke *state* berikutnya tanpa membaca simbol *input*. *NFA* dapat mengirimkan beberapa *state* untuk *input* tertentu.

Demikian laporan ini kami susun dengan harapan adanya aplikasi “Pengurutan Simbol untuk *NFA*” yang kami buat ini dapat memberikan dampak baik bagi siapa pun yang menggunakannya, khususnya para peneliti. Untuk itu kami tertarik membuat aplikasi yang dapat dijadikan media pembelajaran mengenai cara memecahkan masalah dengan baik. Akhir dari penulisan laporan ini, kami mengucapkan terima kasih kepada semua pihak yang telah ikut serta berpartisipasi dalam penyusunan laporan dan kami berharap agar aplikasi ini dapat berjalan dengan baik dan lancar seperti harapan kami.

## Saran

Secara umum, program aplikasi pengujian ini telah memenuhi tujuan-tujuan awal perancangan. Namun masih banyak kemungkinan pengembangan dan penyempurnaan yang bisa dilakukan ke depannya. Adapun beberapa saran penulis sebagai berikut:

1. Menggunakan program perancangan lain yang lebih modern.
2. Menggunakan tampilan yang lebih menarik, misalnya terdapat animasi, multimedia, dan sebagainya.
3. Menggunakan tampilan perancangan dalam bentuk web yang menarik.

# DAFTAR PUSTAKA

Anggraini, I. (2018, 11 9). *APLIKASI PRINT SCREEN*. Diambil kembali dari iinanggraini0809gmail.wordpress.com: https://iinanggraini0809gmail.wordpress.com/2018/11/09/aplikasi-print-screen/

Madjidu, A. (2014, 10 7). *ASCII - AWALUDIN MADJIDU*. Diambil kembali dari mahasiswa.ung.ac.id: https://mahasiswa.ung.ac.id/531414054/home/2014/10/7/ascii.html

S, A. (2020, November 10). *Pengantar Teori Bahasa dan Automata*. Diambil kembali dari alvincahya48.medium.com: https://alvincahya48.medium.com/pengantar-teori-bahasa-dan-a-88bc733f8930

serlyanggraini98. (2018, 11 10). *ANALISIS SISTEM & KEBUTUHAN*. Diambil kembali dari serlyanggraini98.wordpress.com: https://serlyanggraini98.wordpress.com/2018/11/10/analisis-sisitem-kebutuhan/

Trivusi. (2022, September 17). *Finite State Automata: Pengertian, Cara Kerja, dan Jenis-jenisnya*. Diambil kembali dari www.trivusi.web.id: https://www.trivusi.web.id/2022/08/finite-state-automata.html

# LAMPIRAN

## Source Code

#include<iostream>

#include <bits/stdc++.h>

using namespace std;

void sortStringAsc(string s)

{

int N = s.length();

vector<int> freq(256, 0);

for (int i = 0; i < N; i++)

{

freq[s[i]]++;

}

s = "";

for (int i = 0; i < 256; i++)

{

for (int j = 0; j < freq[i]; j++)

{

s = s + (char)i;

}

}

cout<<"Hasil: "<<s<<endl;

return;

}

void sortStringDesc(string s)

{

int N = s.length();

vector<int> freq(256, 0);

for (int i = 0; i < N; i++)

{

freq[s[i]]++;

}

s = "";

for (int i = 256; i > 0; i--)

{

for (int j = 0; j < freq[i]; j++)

{

s = s + (char)i;

}

}

cout<<"Hasil: "<<s<<endl;

return;

}

int main()

{

int pilih;

char lagi;

string input;

do {

system("cls");

cout<<"------------------------------"<<endl;

cout<<"| Aplikasi pengurutan simbol |"<<endl;

cout<<"------------------------------"<<endl;

cout<<"1. Pengurutan Ascending (Menaik)"<<endl;

cout<<"2. Pengurutan Descending (Menurun)"<<endl;

cout<<"------------------------------"<<endl;

cout<<"Pilih menu di atas: ";

cin>>pilih;

cout<<"------------------------------"<<endl;

cout<<"Masukkan simbol-simbol: "<<endl;

cin>>input;

switch(pilih) {

case 1:

sortStringAsc(input);

break;

case 2:

sortStringDesc(input);

break;

default:

break;

}

cout << "Menghitung lagi? Y/t"<<endl;

cin >> lagi;

} while (lagi == 'y' || lagi == 'Y');

cout<<"------------------------------"<<endl;

cout<<"Terima kasih telah menggunakan aplikasi pengurutan simbol"<<endl;

system("pause");

}