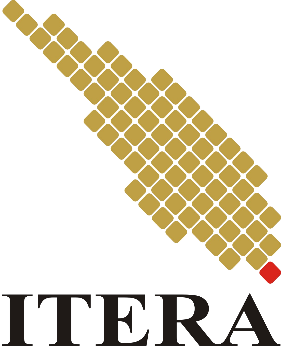
**LAPORAN TUGAS BESAR TEORI BAHASA FORMAL DAN OTOMATA PEMBUATAN COMPILER ATAU INTERPRETER SEDERHANA MENGGUNAKAN BAHASA PEMRROGRAMAN**



**DISUSUN OLEH:**

**BAGAS PANGESTU (14117026)**

**NARDIYANSAH TRI JATMIKO (14117037)**

**DION TRIWIJAYA (16117036)**

**PROGRAM STUDI : TEKNIK INFORMATIKA**

**KELAS : TEORI BAHASA FORMAL DAN OTOMATA - RA**

**DOSEN PENGAMPU : Rahman Indra Kesuma, S.Kom., M.Cs.**

**LAMPUNG SELATAN, 27 MEI 2019**

**INSTITUT TEKNOLOGI SUMATERA**

**2019**

# **KATA PENGEANTAR**

Puji syukur Tim Penyusun panjatkan kepada Allah SWT atas selesainya Laporan Tugas Besar kedua Teori Bahasa Formal dan Otomata, Jurusan Teknik Informatika, Institut Teknologi Sumatera. Penulisan laporan dengan judul “**Pembuatan *Compiler* atau *Interpreter* sederhana Menggunakan Bahasa Pemrograman**” ini memiliki tujuan utama yaitu untuk mengimplementasikan dasar–dasar konsep Teori Bahasa Formal dan *Otomata* kedalam bentuk Program khususnya yang mampu memproses bahasa *Context Free*.

Dalam penulisan laporan ini, Tim Penyusun telah banyak mendapat bimbingan dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini Tim Penyusun mengucapkan terima kasih kepada:

1. Allah SWT. Atas berkah dan rahmat Allah SWT, kami dapat menyelesaikan tugas besar **Pembuatan *Compiler* atau *Interpreter* sederhana Menggunakan Bahasa Pemrograman**.
2. Dosen untuk Pak **Rahman Indra Kesuma, S.Kom., M.Cs.** (sebagai Dosen Pengampu) serta Orang tua yang selalu mendoakan kami.
3. Teman – teman yang telah memberikan arahan, masukan, dalam proses pembuatan Interpreter ini.

Tim Penyusun berharap laporan ini dapat menjelaskan pentingnya pengimplementasian dasar-dasar konsep Teori Bahasa Formal dan *Otomata* kedalam bentuk Program yang baik guna menciptakan sebuah program yang baik dan efisien. Semoga laporan ini dapat memberi manfaat, terima kasih.

Lampung Selatan, 27 Mei 2019

Tim Penyusun

**DAFTAR ISI**

Kata Pengantar1

Bab I Pendahuluan3

1.1 Latar Belakang3

1.2 Batasan Masalah3

1.3 Manfaat Program4

1.4 Tujuan Pembuatan4

1.5 Sistem Penulisan5

Bab II Metodologi6

2.1 Analisis Program6

2.1.1 Pengimplementasian Ide dan Gagasan6

2.1.1 Proses Pembuatan7

2.2 Penulisan Algoritma……………………………………………………….. 8

Bab III Hasil dan Pembahasan9

3.1 Penjelasan Program Beserta Capture Hasil Program9

3.2 Kelebihan dan Kekurangan Program10

3.2.1 Kelebihan Program10

3.2.2 Kekuangan Program10

3.3 Source Code11

Bab IV Kesimpulan dan Saran23

4.1 Kesimpulan23

4.2 Saran23

DAFTAR PUSTAKA24

**BAB I**

**PENDAHULUAN**

* 1. **LATAR BELAKANG**

*Turing machine* adalah mesin abstrak yang mampu mengolah bahasa bebas (*free language*) sehingga menjadi lebih daripada mesin FA(*Finite Automata*) atau PDA(*Push Down Automata*). Turing machine sendiri dapat mengolah bahasa dengan gerakan yang lebih fleksibel yaitu ke kiri dan ke kanan, dan adanya stack menjadi kelebihan mesin ini. Dalam merealisasikan maka dibuatlah tugas besar ini dengan mengambil kasus kalkulator.

Kalkulator adalah mesin perhitungan aritmatika yang dalam tugas besar ini akan ditambahkan di *compiler* yang sudah dibuat di tugas besar pertama. Dalam mesin kalkulator sendiri harus dapat menentukan operasi yang mana yang harus didahulukan, “( )” lebih tinggi dari “\* /”, dan “\* /” lebih tinggi dari “+ -“. Dengan menggunakan *turing machine* diharapkan pembuatan kalkulator menjadi lebih sederhana.

**1.2 BATASAN MASALAH**

Batasan masalah dari compiler ini adalah:

1. Hanya dapat menerima kode “Cetak” dan “Selesai”. Dimana kode “Cetak” untuk menampilkan output kelayar, sedangkan kode “Selesai” untuk keluar dari program.
2. Hanya menggunakan bahasa *C++* dan tidak menggunakan bahasa pemrograman lainnya.
3. Hanya dapat mencetak kalimat dalam tanda petik dan operasi hitung aritmatika sederhana.
   1. **MANFAAT PROGRAM**

Adapun manfaat dari pembuatan program ini yaitu sebagai berikut :

1. Untuk memudahkan pengguna dalam mencari hasil perhitungan,
2. Dengan program ini dapat menjadi inovasi yang menarik dalam sebuah program operasi hitung aritmatika.

## TUJUAN PEMBUATAN

Tujuan pembuatan tugas besar ini adalah sebagai berikut :

1. Mengimplementasikan *Turing machine*

2. Mengerti penggunaan *stack* dalam *Turing machine*

3. Memenuhi tugas besar mata kuliah Teori Bahasa Formal dan Automata

## SISTEMATIKA PENULISAN

Adapun sistematika penulisan dari Laporan Tugas Besar kedua **Pembuatan *Compiler* atau *Interpreter* Sederhana Menggunakan Bahasa**

**Pemrograman** sebagai berikut :

1. **BAGIAN PENDAHULUAN**
   1. Judul
   2. Kata Pengantar
   3. Daftar Isi
   4. Latar Belakang
   5. Batasan Masalah
   6. Manfaat Program
   7. Tujuan Pembuatan
   8. Sistematika Penulisan
2. **BAGIAN ISI**

2.1 Analisis Program

2.1.1 Pengimplementasian Ide dan Gagasan

2.1.2 Proses Pembuatan

2.2 Penulisan Algoritma (Dalam Turing Machine)

2.3 Penjelasan Perogram beserta Capture hasil Program

2.4 Kelebihan dan Kekurangan Program

2.4.1 Kelebihan Program

2.4.2 Kekurangan Program

2.5 *Source Code* Program.

1. **BAGIAN AKHIR**
   1. Kesimpulan
   2. Saran
   3. Daftar Pustaka

**BAB II METODOLOGI**

**2.1 ANALISIS PROGRAM**

**2.1.1 PENGIMPLEMENTASIAN IDE DAN GAGASAN**

Tugas besar kedua ini sebenarnya melanjutkan dari tugas besar pertama mengenai implementasi *DFA*, namun pada tugas besar kedua ide komputasi yang digunakan adalah *Context Free Language*, dimana tiap alur pemerosesan string menggunakan pita dan komputasi ini membutuhkan memori, oleh karena itu tim penyusun setuju menggunakan konsep Turing Machine dan mengimplementasikan dalam bahasa pemerograman C++.

Tim penyusun menggunakan konsep Turing Machine karena *Turing Machine* adalah model komputasi teoritis yang berfungsi sebagai model ideal untuk melakukan perhitungan matematis. Sebuah Mesin *Turing* terdiri atas barisan sel tersusun berupa pita yang dapat bergera maju mundur akan tetapi dalam bergerak maju mundur tersebut secara bertahap atau *step-by-step*. Pada setiap langkah dalam komputasi, mesin ini akan dapat merubah isi dari sel yang aktif pada pita, serta bagaimana menggerakkan pita tersebut.

Konsep yang tim pengembang buat ialah dimana *inputan* pengguna (berupa sintaks) harus sesuai dengan ketentuan *kompiler*, dimana ketentuannya berupa :

**# cetak “sesuatu”; atau # CETAK “SESUATU”;**

*Output* berupa :

* SESUATU

# **cetak 4-2\*4;**

*Output* berupa :

* -4

ketika tiap *state* tidak sesuai dengan apa yang diminta maka akan terjadi *stuck* dan sistem akan memberikan keluaran berupa :

* **masukan anda tidak sesuai dengan format**

untuk lebih jelas tim penyusun bahas di bagian penulisan algoritma.

Untuk membuat program tersebut Tim Penyusun menggunakan *software* berupa *Dev C++* dan *Codeblocks*. Tim Penyusun mengimplementasikan program tersebut dengan memanfaatkan *input* dari pengguna yang kemudian akan diproses menjadi output berupa data.

**2.1.2 PROSES PEMBUATAN**

Dalam pembuatan *compiler* ini kita memerlukan beberapa tahapan yang harus dilakukan, diantaranya adalah:

* 1. Melihat spesifikasi tugas, dengan melihat spesifikasi tugas kita dapat memiliki gambaran compiler seperti apa yang harus dibuat.
  2. Membuat bagan DFA, sebagai blueprint bagaimana suatu compiler memroses sejumlah string.
  3. Mengimplementasikan ke kode, langkah ini dilakukan untuk membuat *interpreter compiler*.

**2.2 PENULISAN ALGORITMA (Menggunakan Bahasa Context Free)**

**1**. Mengecek apakah karakter yang dibaca adalah angka atau operator aritmatika jika angka, maka disimpan dulu dan lanjut mengecek karakter selanjutnya. Jika operator maka simpan di dalam *stack* untuk operator.

**2**. jika karakter berikutnya masih angka maka angka sebelumnya dikalikan dengan 10 dan ditambah angka yang baru dibaca. Lakukan terus sampai bertemu operator aritmatika. Angka akan disimpan di *stack* untuk angka.

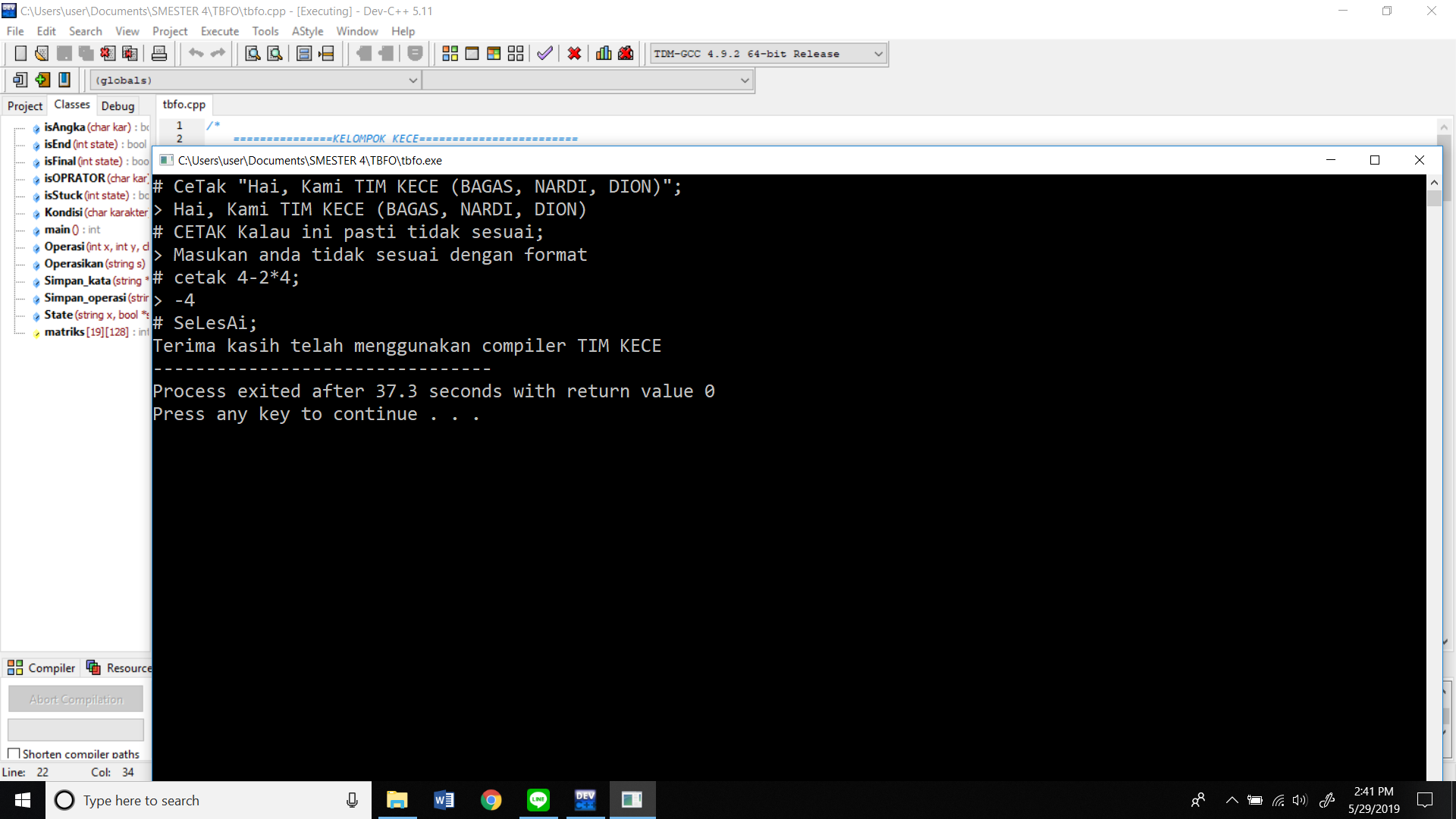
**3**. jika ada operator yang ingin masuk maka angka dan operator yang ada didalam stack dioperasikan terlebih dahulu.

**4**. jika dibaca karakter ‘(‘ maka operator setelahnya disimpan dulu dan jika setelahnya dibaca karakter ‘)’ maka berjalan mundur di kedua *stack* sampai di *stack* operator ditemukan karakter ‘(‘.

**5**. urutan operator di stack operator disusun berdasarkan tingkatan operator aritmatika, yang berarti operator dengan tingkat paling tinggi berada di *top stack* yang berarti operator tersebut akan dikerjakan lebih dahulu. Urutan operator dari tertinggi ke terendah sebagai berikut (,),\*,/,+,-.

**BAB III HASIL DAN PEMBAHASAN**

## PENJELASAN PROGRAM BESERTA CAPTURE HASIL PROGRAM



**Penjelasan :**

Program hanya menerima sintaks :

*CETAK “isi\_sendiri”;* (kata cetak bersifat *non-sensitive case*).

Inputan diawali simbol “*#*” dan pengguna memasukkan sintaks yang sesuai. Ketika pengguna selesai menginput ketik enter untuk menampilkan keluaran yang diawali simbol “*>*” jika input yang dimasukkan tidak sesuai sintaks maka pengguna akan menerima keluaran “*syntax error*”.

sintaks untuk selesai adalah sebagai berikut :

Selesai; (kata Selesai bersifat *non-sensitive case*).

CETAK 4-2\*4; (Kata cetak bersifat *non sensitive case*,simbol operasi aritmatika harus berada diantara angka).

Ketika pengguna memasukkan CETAK 4-2\*4; maka yang pertama di proses adalah simbol \* (perkalian) kemudian melakukan operasi perkalian (-2\*4 = -8), setelah itu simbol – (pengurangan) diproses (4-8 = -4), ini dilakukan berdasarkan ide komputasi *Context Free*.

## KEKURANGAN DAN KELEBIHAN PROGRAM

**3.2.1 KEKURANGAN PROGRAM**

Program ini masih memiliki beberapa kekurangan sebagai berikut :

1. masih menggunakan tipe integer yang berarti tidak bisa menghasilkan bilangan pecahan.

2. operator aritmatika yang digunakan masih sedikit.

3. Hanya dapat memproses sintaks “Cetak” dan “Selesai”,

4. Hanya dapat mencetak kalimat dalam tanda petik dan operasi hitung aritmatika sederhana,

**3.2.2 KELEBIHAN PROGRAM**

1. Program menggunakan konsep Turing machine pada operasi aritmatika.

2. Progam yang dihasilkan sesuai dengan spesifikasi kebutuhan tugas besar.

3. Tidak ada error atau bug pada program ketika menginputkan sesuai perintah.

**3.3 *SOURCE CODE***

/\*

===============KELOMPOK KECE========================

ANGGOTA : BAGAS PANGESTU (14117026)

NARDIYANSAH TRI JATMIKO (14117037)

DION TRIWIJAYA (16117036)

\*/

**#include <bits/stdc++.h>** // library yang mencakup seluruh library standar seperti iostream, stdio, stdlib, stack dll.

**using namespace std;**

//Definisi matriks secara Global agar tidak perlu inisiasi lagi pada sub - program

//matriks[19][128] maksudnya adalah total state ada 19 dan jumlah karakter ASCII ada 128

**int matriks[19][128];**

//memastikan yang dimasukkan adalah digit angka

**bool isAngka(char kar){**

**return (kar >= '0' && kar <='9');**

**}**

//operator - operator

**bool isOPRATOR(char kar){**

**return(kar=='+' || kar=='-' || kar=='\*' || kar=='/' || kar=='(' || kar==')');**

**}**

//kondisi operasi

**int Kondisi(char karakter){**

**switch(karakter){**

**case '+' :**

**case '-' : return 1;**

**case '\*' :**

**case '/' : return 2;**

**case '(' :**

**case ')' : return 3;**

**default : return -1;**

**}**

**}**

//ketika menemukan simbol pada inputan maka yang dilakukan adalah seperti dibawah

**int Operasi(int x, int y, char c){**

**if(c=='+'){**

**return x+y;**

**}else if(c=='-'){**

**return x-y;**

**}else if(c=='\*'){**

**return x\*y;**

**}else{**

**return x/y;**

**}}**

//operasi aritmatika

**int Operasikan(string s){**

**stack<int> XNILAI;** // menggunakan stack pada implementasi program

**stack<char> OPERASI;**

**int temp=0;**

**int i=0;**

**while(i<s.length()){**

**char XKAR = s[i];**

**if(isAngka(XKAR)){**

**temp= (temp\*10) + (int)(XKAR - '0');**

**}else if(isOPRATOR(XKAR)){**

**if(XKAR=='('){**

**OPERASI.push(XKAR);**

**temp=0;**

**}else if(XNILAI.empty()){** // jika kosong nilainya

**XNILAI.push(temp);**

**OPERASI.push(XKAR);**

**temp=0;**

**}else if(XKAR==')'){**

**XNILAI.push(temp);**

**while(OPERASI.top() != '('){**

**XKAR=OPERASI.top();**

**OPERASI.pop();**

**temp=XNILAI.top();**

**XNILAI.pop();**

**int pre = XNILAI.top();**

**XNILAI.pop();**

**temp = Operasi(pre,temp,XKAR);**

**XNILAI.push(temp);**

**}**

**OPERASI.pop();**

**XNILAI.pop();**

**}else{**

**char prec = OPERASI.top();**

**if(prec=='('){**

**XNILAI.push(temp);**

**OPERASI.push(XKAR);**

**temp=0;**

**}else if(Kondisi(XKAR) > Kondisi(prec)){**

**XNILAI.push(temp);**

**OPERASI.push(XKAR);**

**temp=0;**

**}else{**

**int PREVNILAI = XNILAI.top(); XNILAI.pop();**

**char prevop = OPERASI.top();**

**OPERASI.pop();**

**PREVNILAI=Operasi(PREVNILAI,temp,prevop);**

**XNILAI.push(PREVNILAI);**

**OPERASI.push(XKAR);**

**temp=0;**

**}**

**}**

**}**

**i++;**

**}**

**while(!OPERASI.empty()){**

**int prev = XNILAI.top();**

**XNILAI.pop();**

**char preop = OPERASI.top();**

**OPERASI.pop();**

**temp = Operasi(prev,temp,preop);**

**}**

**return temp;**

**}**

**bool isStuck(int state){**

**if (state < 0){**

**return true;**

**}else{**

**return false;**

**}**

**}**

**bool isFinal(int state){**

**if(state == 9 || state == 17 || 19){**

**return true;**

**}else{**

**return false;**

**}**

**}**

**bool isEnd(int state){**

**if(state == 17){**

**return true;**

**}else{**

**return false;**

**}**

**}**

**void Simpan\_operasi (string \*x, char y){**

**\*x = \*x + y;**

**}**

**void Simpan\_kata (string \*x, char y){**

**\*x = \*x + y;**

**}**

//menelusuri state

**void State(string x, bool \*selesai){**

**int i = 0;**

**int state = 0;**

**string kata="\0";**

**string oper="\0";**

**while(i < x.length() && isStuck(state) == false){**

**state = matriks[state][x[i]];**

**if(state == 7 && x[i] != 34){**

**Simpan\_kata(&kata, x[i]);**

**}else if(state==18 && x[i] != 59){**

**Simpan\_operasi(&oper,x[i]);**

**}**

**i++;**

**}**

**if(isStuck(state) == true || isFinal(state) == false ){**

**kata = "\0";**

**cout << "> Masukan anda tidak sesuai dengan format"<< endl;**

**return;**

**}else{**

**if(isEnd(state) == true){**

**cout << "Terima kasih telah menggunakan compiler TIM KECE";**

**\*selesai = true;**

**}else if(state==19){**

**int temp = Operasikan(oper);**

**cout << "> " << temp << endl;**

**}else**

**cout << "> " << kata << endl;**

**return;**

**}**

**}**

**int main(){**

**for(int i = 0 ; i < 19 ; i++){**

**for(int j = 0 ; j < 128 ; j++){**

**matriks[i][j] = -1;**

**}**

**}**

//membuat table transisi state

**matriks[0][32] = 0;**

//huruf kecil

**matriks[0][99] = 1;**

**matriks[1][101] = 2;**

**matriks[2][116] = 3;**

**matriks[3][97] = 4;**

**matriks[4][107] = 5;**

**matriks[5][32] = 6;**

**matriks[6][32] = 6;**

**matriks[6][34] = 7;**

//operator aritmatika dan angka

**matriks[6][40]=18;**

**matriks[6][41]=18;**

**matriks[6][42]=18;**

**matriks[6][43]=18;**

**matriks[6][45]=18;**

**matriks[6][47]=18;**

**matriks[6][48]=18;**

**matriks[6][49]=18;**

**matriks[6][50]=18;**

**matriks[6][51]=18;**

**matriks[6][52]=18;**

**matriks[6][53]=18;**

**matriks[6][54]=18;**

**matriks[6][55]=18;**

**matriks[6][56]=18;**

**matriks[6][57]=18;**

**matriks[18][40]=18;**

**matriks[18][41]=18;**

**matriks[18][42]=18;**

**matriks[18][43]=18;**

**matriks[18][45]=18;**

**matriks[18][47]=18;**

**matriks[18][48]=18;**

**matriks[18][49]=18;**

**matriks[18][50]=18;**

**matriks[18][51]=18;**

**matriks[18][52]=18;**

**matriks[18][53]=18;**

**matriks[18][54]=18;**

**matriks[18][55]=18;**

**matriks[18][56]=18;**

**matriks[18][57]=18;**

**matriks[18][32]=18;**

**matriks[18][59]=19;**

//huruf besar

**matriks[0][67] = 1;**

**matriks[1][69] = 2;**

**matriks[2][84] = 3;**

**matriks[3][65] = 4;**

**matriks[4][75] = 5;**

**for(int i = 0 ; i < 128 ; i++){**

**matriks[7][i] = 7;**

**}**

**matriks[7][34] = 8;**

**matriks[8][32] = 8;**

**matriks[8][59] = 9;**

**matriks[0][115] = 10;**

**matriks[10][101] = 11;**

**matriks[11][108] = 12;**

**matriks[12][101] = 13;**

**matriks[13][115] = 14;**

**matriks[14][97] = 15;**

**matriks[15][105] = 16;**

**matriks[16][32] = 16;**

**matriks[16][59] = 17;**

**matriks[0][83] = 10;**

**matriks[10][69] = 11;**

**matriks[11][76] = 12;**

**matriks[12][69] = 13;**

**matriks[13][83] = 14;**

**matriks[14][65] = 15;**

**matriks[15][73] = 16;**

//IO pada program

**string x;**

**bool selesai = false;**

**while(selesai ==false){**

**cout <<"# ";**

**getline(cin,x);**

**State(x, &selesai);**

**}**

**}**

**BAB IV KESIMPULAN DAN SARAN**

**4.1 KESIMPULAN**

Teori Bahasa Formal dan Otomata mempelajari tentang bagaimana suatu mesin berjalan dengan berpindah dari satu state ke state yang lain. Tugas ini mengimplementasikan teori tersebut pada intrepeter compiler sederhana. *Compiler* berpindah dari satu *state* ke *state* yang lain dengan memproses setiap karakter yang ada apakah sesuai dengan aturan penulisan sintaks yang diterima. Dengan demikian pembuatan compiler merupakan salah satu penerapan dari ilmu yang didapat di matakuliah teori bahasa formal dan otomata.

**4.2 SARAN**

Masih disadari bahwa program ini masih memilik beberapa kekurangan, untuk selanjutnya program ini bisa menggunakan tipe data *float* agar dapat menghasilkan bilangan pecahan, dan operator aritmatika yang digunakan bisa lebih banyak lagi.

**DAFTAR PUSTAKA**

[*https://id.wikipedia.org/wiki/Mesin\_Turing*](https://id.wikipedia.org/wiki/Mesin_Turing)

*Slide Power Point Teori Bahasa Formal dan Otomata.*