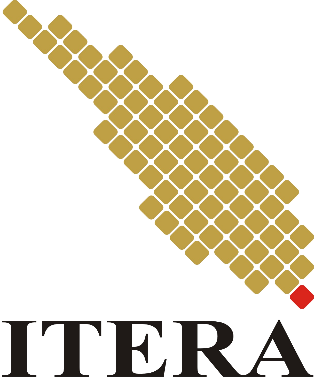
****

LAPORAN TUGAS BESAR

TEORI BAHASA FORMAL DAN OTOMATA

Disusun Oleh :

IRMA SAFITRI (14117067)

NURUL FAUZIA AZIZAH (14117071)

NURTIAS RAHAYU (14117086)

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN SAINS

INSTITUT TEKNOLGI SUMATERA

LAMPUNG SELATAN

2019

# **KATA PENGANTAR**

Puji syukur atas kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penyusun dapat menyelesaikan Penulisan Laporan Tugas Besar dengan tema “INTERPRETER SEDERHANA” dengan tepat waktu. Diharapkan dengan dibuatnya laporan ini khhususnya para pembaca dapat menambah wawasan dan ilmu pengetahuan mengenai pemrograman komputer.

Kami mengucapkan terima kasih kepada Bpk. Angga Wijaya, S. Si., M.Si dan Ibu Dr. Massayu Leyla Khodra.S.T., M.T selaku dosen pengampu matakuliah Teori Bahasa Formal dan Otomata, serta rekan – rekan yang telah membantu dalam pembuatan *programming project* dan penyusunan laporan Tugas Besar.

Mengingat bahwa laporan ini masih jauh dari kata sempurna, oleh karena itu penyusun mengharapkan segala bentuk kritikan dan saran yang membangun dari pembaca guna meningkatkan kualitas dari karya tulis ini.

Penyusun juga mengucapkan terima kasih kepada seluruh pihak yang telah berperan dalam penyusunan laporan ini. Semoga Allah senantiasa meridhoi usaha yang telah dilakukan.

Lampung Selatan, 20 Mei 2019

Penyusun

**DAFTAR ISI**

[**KATA PENGANTAR** 1](#_Toc10050469)

[**BAB I PENDAHULUAN** 3](#_Toc10050470)

[1.1 Latar Belakang 3](#_Toc10050471)

[1.2 Batasan Masalah 4](#_Toc10050472)

[1.3 Tujuan 4](#_Toc10050473)

[1.4 Batasan Masalah 4](#_Toc10050474)

[**BAB IIDASAR TEORI** 6](#_Toc10050475)

[**BAB III HASIL DAN PEMBAHASAN** 12](#_Toc10050476)

[3.1 Algoritma dari Program Interpreter Sederhana 12](#_Toc10050477)

[3.2 Output 16](#_Toc10050478)

[3.3 Rancangan Mesin Turing 17](#_Toc10050479)

[**BAB IV KESIMPULAN DAN SARAN** 19](#_Toc10050480)

[4.1 Kesimpulan 19](#_Toc10050481)

[4.2 Saran 19](#_Toc10050482)

# **BAB I PENDAHULUAN**

## Latar Belakang

Bahasa alami atau bahasa non formal adalah bahasa komunikasi yang digunakan oleh manusia, yaitu bahasa ucap seperti bahasa Indonesia, bahasa Inggris, bahasa Spanyol, dan bahsa-bahasa lainnya. Sintaks bahasa alami sangat rumit, karena bahasa alami memiliki tata bahasa dan aturan yang lebih luas sehingga tidak mungkin untuk menspesifikasi semua aturan sintaksnya. Bahasa formal adalah bahasa yang kaidah sintaksnya dapat dispesifikasi secara matematis dengan baik. Kaidah bahasa dalam teori bahasa formal tidak hanya bermanfaat untuk studi linguistik bahasa alai seperti penerjemahan secara otomatis dari suatu bahasa ke bahasa lain, tetapi juga berguna untuk studi bahasa pemrograman.

Sedangkan teori otomata adalah teori mengenai mesin-mesin abstrak, dan berkaitan erat dengan teori bahasa formal. Otomata adalah mesin abstrak yang dapat mengenali, menerima, atau membangkitkan sebuah kalimat dalam bahasa tertentu. Otomata merupakann suatu sistem yang terdiri atas sejumlah berhingga state, dimana state menyatakan informasi mengenai input yang dianggap sebagai bahasa yang harus dikenali mesin yang selanjutnya akan diterima atau tidak.

Dengan menggunakan teori bahasa formal dan otomata kita dapat membuat program-program yang dapat membantu pekerjaan manusia. Salah satunya adalah interpreter. Interpreter adalah perangkat lunak yang mampu mengeksekusikan code program lalu menerjemahaknnya ke dalam bahasa mesin, sehingga mesin melakukan instruksi yang diminta oleh programmer tersebut.

Dengan menggunakan teori bahasa formal dan otomata kita dapat membuat program-program yang dapat membantu pekerjaan manusia. Salah satunya adalah kalkulator. Kalkulator adalah salah satu alat yang digunakan untuk membantu mengkalkulasikan angka-angka. Alat ini dibuat agar memudahkan serta mempercepat dalam melakukan pekerjaan yang erat kaitannya dengan perhitungan. Sehingga, kalkulator hampir selalu digunakan di segala aspek kehidupan yang berhubungan dengan angka-angka. Pada era modern ini, tidak hanya dalam bentuk fisik, kalkulator juga tersedia dalam bentuk perangkat lunak seperti kalkulator di komputer dan *handphone.* Kita dapat dengan mudah mengunduh kalkulator dengan mudah pada perangkat tersebut. Kalkulator dapat dibuat sesuai kebutuhan masing-masing selama orang tersebut mampu dan memahami bahasa pemrograman untuk membuat sebuah perangkat lunak.

## Batasan Masalah

Pada penelitian ini perlu didefinisikan beberapa batasan sejauh mana penelitian ini akan dikerjakan. Batasan-batasan masalah tersebut adalah sebagai berikut:

1. Bagaimana cara perancangan pembuatan interpreter sederhana ini ?
2. Bagaimana cara pembuatan program interpreter sederhana ini?
3. Bagaimana cara implementasi interpreter sederhana tersebut ?

## Tujuan

Adapun tujuan dari pembuatan interpreter sederhana ini adalah sebagai berikut

1. Untuk mengetahui cara perancangan interpreter sederhana yang memanfaatkan teori bahasa formal dan otomata.
2. Untuk mengetahui cara pembuatan program interpreter sederhana.
3. Untuk mengetahui cara implementasi interpreter sederhana tersebut.

## 1.4 Batasan Masalah

* 1. Membuat interpreter sederhana yang akan menghasilkan luaran dengan menerima string “ CETAK<spasi>“<string luaran>”; ” dan “ CETAK<spasi><rangkaian ekspresi aritmatika>; ”
  2. Interpreter dapat mendeteksi penerimaan maupun penolakan string dengan melakukan pengecekan pada setiap karakter yang di inputkan oleh pengguna.
  3. Untuk kasus string yang diterima, program mampu mengeluarkan keluaran berupa ‘string luaran’ yang telah dimasukkan.
  4. Pada proses aritmatika, untuk kasus string diterima, program mampu mengeluarkan keluaran berupa hasil dari perhitungan ‘rangkaian ekspresi aritmatika’ yang telah dimasukkan.
  5. Untuk kasus string yang tidak diterima, program memunculkan tulisan “Kode tidak sesuai dengan peraturan”.

# **BAB IIDASAR TEORI**

TEORI BAHASA

Teori bahasa membicarakan bahasa formal, terutama untuk kepentinga perancangan kompilator (*compiler*) dan pemroses naskah.

* 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’ disebut ‘bahasa’ saja.

AUTOMATA

Automata adalah mesin abstrak yang dapat mengenali, menerima, atau membangkitkan sebuah kalimat dalam bahasa tertentu.

Untuk memodelkan hardware dari komputer diperkenalkan otomata. Otomata adalah fungsi-fungsi dari komputer digital. Menerima *input*, mengh asilkan *output*, bisa memiliki penyimpanan sementara dan mampu membuat keputusan dalam mentransformasikan *input* ke *output*.

Sebuah bahasa formal adalah suatu abstraksi terdiri dari himpunan simbol-simbol dan aturan-aturan yang mana simbol-simbol tersebut bisa dikombinaasikan ke dalam entitas yang disebut kalimat.

Meskipun bahasa formal yang dipelajari disini lebih sederhana daripada bahasa lebih sederhana daripada bahasa pemrograman, meraka mempunyai banyak hal yang penting. Kita bisa mempelajari banyak tentang bahasa pemrograman dari bahasa formal.

Otomata merupakan suatu sistem yang terdiri atas sejumlah berhingga state, dimana state menyatakan informasi mengenai input yan lalu dandapaty dianggap sebagai memory mesin.

Input pada mesin otomata dianggap sebagai bahasa yang harus dikenali oleh mesin. Selanjutnya, mesin otomata membuat bkeputusan yang mengindikasikan apakah input itu diterima atau tidak.

BEBERAPA PENGERTIAN DASAR

Simbol adalah sebuah entitas abstrak (seperti halnya pengertian titik dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol. Pada umumnya kita menggunakan huruf kecil (lower case)atau angka untuk melambngkan simbol, dan huruf kecil diakhir alphabet khususnya w,x,y,z untuk melambangkan untai (String).

String adalah deretan terbatas (finite) simbol-simbol. Sebagai contoh, jika a,b dan c adalah tiga buah simbol maka abcb adalah sebuah String yang dibangun dari ketiga simbol tersebut.

Jika w adalah sebuah String maka panjang String dinyatakan sebagai |w| dan didefinisikan sebagai cacahan (banyaknya) simbol yang menyusun String tersebut. Sebagai contoh, jika w=abcb maka |w|=4.

String hampa adalah sebuah String dengan nol buah simbol. Dinyatakan dengan simbol Ɛ (atau ˄) sehingga | Ɛ|=0. String hampa dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.

Alphabet adalah himpunan hingga (finite set) simbol-simbol. Pada umumnya kita menggunakan lower case untuk melambangkan simbol, dan huruf  kecil diakhir alphabet khususnya w, x, y dan z untuk melambangkan untai (String).

OPERASI DASAR STRING

Diberikan dua String : x=abc , dan y=123;

* PREFIX STRING w adalah String yang dihasilkan dari String w dengan menghilangkan *nol* atau lebih simbol-simbol *paling belakang* dari String w tersebut. Contoh : abc, ab, a dan Ɛ adalah semua PREFIX(x).
* PROPERPREFIK STRING w adalah String yang dihasilkan dari String w dengan menghilangkan *satu*atau lebih simbol-simbol *paling* *belakang* dari String w tersebut. Contoh : ab, a dan Ɛ adalah semua PROPERPREFIX(x).
* POSTFIX (atau SUFIX) STRING w adalah String yang dihasilkan dari String w dengan menghilangkan *nol* atau lebih simbol-simbol *paling* *depan* dari String w tersebut. Contoh : abc, bc, c dan Ɛ adalah semua POSTFIX(x).
* PROPERPOSTFIX (atau SUFIX) STRING w adalah String yang dihasilkan dari String w dengan menghilangkan *satu* atau lebih simbol-simbol *paling* *depan* dari String w tersebut. Contoh : bc, c dan Ɛ adalah semua PROPERPOSTFIX(x).
* SUBSTRING STRING w adalah  String yang dihasilkan dari String w dengan menghilangkan *nol* atau lebih simbol *paling* *depan* *dan/atau simbol-simbol paling belakang* dari String w tersebut. Contoh : abc, ab, bc, a, b, c dan Ɛ adalah semua SUBSTRING (x).
* PROPER SUBSTRING STRING w adalah  String yang dihasilkan dari String w dengan menghilangkan *satu* atau lebih simbol *paling* *depan* *dan/atau simbol-simbol paling belakang* dari String w tersebut. Contoh : ab, bc, a, b, c dan Ɛ adalah semua PROPERSUBSTRING (x).
* SUBSEQUENCE STRING w adalah  String yang dihasilkan dari String w dengan menghilangkan *nol* atau lebih *simbol-simbol* dari String w tersebut. Contoh : abc, ab, bc, ac, a, b, c dan Ɛ adalah semua SUBSEQUENCE (x).
* PROPER SUBSEQUENCE STRING w adalah  String yang dihasilkan dari String w dengan menghilangkan *satu* atau lebih *simbol-simbol* dari String w tersebut. Contoh : ab, bc, ac, a, b, c dan Ɛ adalah semua SUBSEQUENCE (x).
* HEAD STRING w adalah simbol paling depan dari String w. Contoh : a adalah HEAD(x).
* TAIL STRING w adalah String yang dihasilkan dari String w dengan menhilangkan HEAD tersebut. Contoh : bc adalah TAIL(x).
* CONCATENATION adalah penyambungan dua buah String. Contoh : concate(xy) = xy = abc123.
* ALTERNATION adalah pilihan satu diantara dua buah String. Operatornya adalah |. Contoh : alternate(xy) = x | y = abc atau 123.
* KLEENE CLOSURE : x\* = Ɛ | x | xx | xxx| … = Ɛ | x | x2 | x3| …

x\* : menyatakan himpunan seluruh untai yang meliputi seluruh alphabet, termasuk untai kosong (Ɛ).

* POSITIVE CLOSURE : x+ = x | xx | xxx |… = x | x2 | x3 | …

x+ : menyatakan himpunan seluruh untai meliputi seluruh alphabet, *tidak* termasuk untai kosong (Ɛ).

KONSEP DASAR

Dalam pembicaraan Grammar, anggota alphabet dinamakan simbol terminal atau token.

Kalimat adalah deretan hinggga simbol-simbol terminal.

Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa hingga bisa tak hingga kalimat.

Simbol berikut adalah simbol terminal :

* Huruf kecil awal alphabet, misal a, b, c
* Simbol operator, misal +, -, dan x
* Simbol tanda baca, misal (,), dan ;
* String  yang tercetak tebal, misal **if, then, else**

Simbol berikut adalah simbol non terminal :

* Huruf besar awal alphabet, misal A, B, C
* Huruf S sebagai simbol awal
* String yang tercetak miring, misal *expr* dan *stmt*

* Huruf besar akhir alphabet melambangkan simbol terminal atau non terminal. Misal X, Y, Z
* Huruf kecil akhir alphabet melambangkan String yang tersusun atas simbol-simbol terminal, misal x, y, z
* Huruf yunani melambangkan String yang tersusun atas simbol-simbol terminal atau non terminal atau campuran keduanya, misal α, β, γ
* Sebuah produksi dilambangkan sebagai  α → β artinya dalam sebuah derivasi dapat dilakukan penggantian simbol α dengan simbol β
* Dalam produksi berbentuk α → β.  α disebut ruas kiri sedangkan  β  disebut ruas kanan.
* Derivasi adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi dilambangkan sebagai α ⇒β.
* Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya.
* Kalimat adalah string yang terssusun atas simbol-simbol terminal. Jelaslah bahwa kalimat adalah kasus khusus dari sentensial.
* Pengertian terminal berasal dari kata terminate (berakhir), maksudnya derivasi berakhir jika sentensial yang dihasilkan adalah sebuah kalimat (tersusun atas simbol terminal).
* Pengertian non terminal berasal dari kata not terminate (tidak/belum berhenti), derivasi tidak/belum berakhir jika sentensial yang dihasilkan mengandung simbol non terminal.

GRAMMAR DAN KLASIFIKASI CHOMSKY

Grammar G didefinisikan sebagai pasangan 4 Tuple : VT, VN, S , Q dan dituliskan sebagai G (VT, VN, S, Q), dimana :

* VT : himpunan simbol terminal (atau himpunan token-token, atau alphabet)
* VN : himpunan simbol-simbol non terminal
* S ϵ VN   : simbol awal (atau simbol start)
* Q             : himpunan produksi

Berdasarkan komposisi bentuk ruas kiri dan kanan produksinya (α → β), Noam Chomsky mengklasifikasikan 4 tipe grammar :

1. Grammar tipe-0 : UNRESTRICTED GRAMMAR (UG)

Ciri : α, β ϵ (VT | VN)\*, | α | > 0

1. Grammar tipe-1 : CONTEXT SENSITIVE GRAMMAR (CSG)

Ciri : α, β ϵ (VT | VN)\*, 0 < | α | ≤ | β |

1. Grammar tipe-2 : CONTEXT FREE GRAMMAR (CFG)

Ciri : α ϵ VN , β ϵ (VT | VN)\*

1. Grammar tipe-3 : REGULLAR GRAMMAR (RG)

Ciri : α ϵ VN , β ϵ {VT , VT VN} atau α ϵ VN , β ϵ {VT , VN VT }

Mengingat ketentuan simbol-simbol  maka ciri RG sering ditulis sebagai :

α ϵ VN , β ϵ {a , bC}  atau α ϵ VN , β ϵ {a , Bc}

Contoh analisa penentuan type grammar

1. Grammar G1 dengan Q1 = {S→aB, B→bB, B→b}.

2. Grammar G2 dengan Q2 = {S→Ba, B→bB, B→b}.

Jawab :

1. Grammar G1 dengan Q1 = {S→aB, B→bB, B→b}. Ruas kiri semua produksinya terdiri dari sebuah VN maka G1 kemungkinan type CFG atau RG.

Selanjutnya karena semua ruas kanannya terdiri dari sebuah VT atau String VT VN maka G1 adalah RG

1. Grammar G2 dengan Q2 = {S→Ba, B→bB, B→b}.  Ruas kiri semua produksinya terdiri dari sebuah VN maka G2 kemungkinan type CFG atau RG.

Selanjutnya karena semua ruas kanannya mengandung String VT VN (yaitu bB) dan juga String VN VT (Ba) maka G3 bukan RG, dengan kata lain G3 adalah CFG.

# **BAB III HASIL DAN PEMBAHASAN**

## Algoritma dari Program Interpreter Sederhana

* + - 1. //Irma Safitri (14117067)

1. //Nurul Fauzia Azizah(14117071)
2. //Nurtias Rahayu (14117086)
4. #include <bits/stdc++.h>
5. **using** **namespace** std;
7. **int** matrix[19][128];
8. **void** setNull() {
9. matrix[0][32] = 0;
10. **for**(**int** i = 0 ; i < 128 ; i++){
11. matrix[7][i] = 7;
12. }
14. matrix[0][99] = 1;
15. matrix[1][101] = 2;
16. matrix[2][116] = 3;
17. matrix[3][97] = 4;
18. matrix[4][107] = 5;
20. matrix[0][67] = 1;
21. matrix[1][69] = 2;
22. matrix[2][84] = 3;
23. matrix[3][65] = 4;
24. matrix[4][75] = 5;
26. matrix[5][32] = 6;
27. matrix[6][32] = 6;
28. matrix[6][34] = 7;
30. matrix[6][40]=18;
31. matrix[6][41]=18;
32. matrix[6][42]=18;
33. matrix[6][43]=18;
34. matrix[6][45]=18;
35. matrix[6][47]=18;
36. matrix[6][48]=18;
37. matrix[6][49]=18;
38. matrix[6][50]=18;
39. matrix[6][51]=18;
40. matrix[6][52]=18;
41. matrix[6][53]=18;
42. matrix[6][54]=18;
43. matrix[6][55]=18;
44. matrix[6][56]=18;
45. matrix[6][57]=18;
47. matrix[18][40]=18;
48. matrix[18][41]=18;
49. matrix[18][42]=18;
50. matrix[18][43]=18;
51. matrix[18][45]=18;
52. matrix[18][47]=18;
53. matrix[18][48]=18;
54. matrix[18][49]=18;
55. matrix[18][50]=18;
56. matrix[18][51]=18;
57. matrix[18][52]=18;
58. matrix[18][53]=18;
59. matrix[18][54]=18;
60. matrix[18][55]=18;
61. matrix[18][56]=18;
62. matrix[18][57]=18;
63. matrix[18][32]=18;
64. matrix[18][59]=19;
66. matrix[7][34]=8;
67. matrix[8][32]=8;
68. matrix[8][59]=9;
69. matrix[0][115]=10;
70. matrix[10][101]=11;
71. matrix[11][108]=12;
72. matrix[12][101]=13;
73. matrix[13][115]=14;
74. matrix[14][97]=15;
75. matrix[15][105]=16;
76. matrix[16][32]=16;
77. matrix[16][59]=17;
78. matrix[0][83]=10;
79. matrix[10][69]=11;
80. matrix[11][76]=12;
81. matrix[12][69]=13;
82. matrix[13][83]=14;
83. matrix[14][65]=15;
84. matrix[15][73]=16;
85. }
86. **bool** operasi(**char** cek){
87. **return** ( cek=='+' || cek=='-' || cek=='\*' || cek=='/' || cek=='(' || cek==')' );
88. }
89. **bool** digit(**char** cek){
90. **return** (cek >= '0' && cek <= '9');
91. }
92. **int** perintahoperasi(**int** x, **int** y, **char** z){
93. **if**(z=='+'){
94. **return** x+y;
95. }**else** **if**(z=='-'){
96. **return** x-y;
97. }**else** **if**(z=='\*'){
98. **return** x\*y;
99. }**else**{
100. **return** x/y;
101. }
102. }
103. **int** penting(**char** cek){
104. **switch**(cek){
105. **case** '+' :
106. **case** '-' : **return** 1;
107. **case** '\*' :
108. **case** '/' : **return** 2;
109. **case** '(' :
110. **case** ')' : **return** 3;
111. **default**  : **return** -1;
112. }
113. }
114. **int** init(string inputan){
115. stack<**int**> str;
116. stack<**char**> operatorr;
118. **int** sementara=0;
119. **int** i=0;
120. **while**(i<inputan.length()){
121. **char** karakter = inputan[i];
122. **if**(digit(karakter)){
123. sementara = (sementara\*10) + (**int**)( karakter - '0');
124. }**else** **if**(operasi(karakter)){
125. **if**( karakter=='('){
126. operatorr.push(karakter);
127. sementara=0;
128. }**else** **if**(str.empty()){
129. str.push(sementara);
130. operatorr.push(karakter);
131. sementara=0;
132. }**else** **if**(karakter==')'){
133. str.push(sementara);
134. **while**(operatorr.top() != '('){
135. karakter=operatorr.top();
136. operatorr.pop();
137. sementara=str.top();
138. str.pop();
139. **int** pre = str.top();
140. str.pop();
141. sementara = perintahoperasi(pre,sementara, karakter);
142. str.push(sementara);
143. }
144. operatorr.pop();
145. str.pop();
146. }**else**{
147. **char** prec = operatorr.top();
148. **if**(prec=='('){
149. str.push(sementara);
150. operatorr.push(karakter);
151. sementara=0;
152. }**else** **if**(penting(karakter) > penting(prec)){
153. str.push(sementara);
154. operatorr.push(karakter);
155. sementara=0;
156. }**else**{
157. **int** prevval = str.top();
158. str.pop();
159. **char** prevop = operatorr.top();
160. operatorr.pop();
161. prevval=perintahoperasi(prevval,sementara,prevop);
162. str.push(prevval);
163. operatorr.push(karakter);
164. sementara=0;
165. }
166. }
167. }
168. i++;
169. }
170. **while**(!operatorr.empty()){
171. **int** prev = str.top();
172. str.pop();
173. **char** preop = operatorr.top();
174. operatorr.pop();
175. sementara = perintahoperasi(prev,sementara,preop);
176. }
177. **return** sementara;
178. }
179. **bool** salah(**int** states){
180. **if**(states < 0){
181. **return** **true**;
182. }**else**{
183. **return** **false**;
184. }
185. }
186. **bool** akhir(**int** states){
187. **if**(states == 9 || states == 17 || 19){
188. **return** **true**;
189. }**else**{
190. **return** **false**;
191. }
192. }
193. **bool** berakhir(**int** states){
194. **if**(states == 17){
195. **return** **true**;
196. }**else**{
197. **return** **false**;
198. }
199. }
200. **void** operasiii(string \*x, **char** y){
201. \*x = \*x + y;
202. }
203. **void** kataaa(string \*x, **char** y){
204. \*x = \*x + y;
205. }
206. **void** state(string x, **bool** \*selesai){
207. **int** i = 0;
208. **int** state = 0;
209. string kata="\0";
210. string oper="\0";
212. **while**(i < x.length() && salah(state) == **false**){
213. state = matrix[state][x[i]];
214. **if**(state == 7 && x[i] != 34){
215. kataaa(&kata, x[i]);
216. }**else** **if**(state==18 && x[i] != 59){
217. operasiii(&oper,x[i]);
218. }
219. i++;
220. }
221. **if**(salah(state) == **true** || akhir(state) == **false** ){
222. kata = "\0";
223. cout << "Kode tidak sesuai dengan peraturan\n";
224. **return**;
225. }**else**{
226. **if**(berakhir(state) == **true**){
227. cout << "Terimakasih sudah menggunakan compiler ini";
228. \*selesai = **true**;
229. }**else** **if**(state==19){
230. **int** sementara = init(oper);
231. cout << "> " << sementara << endl;
232. }**else**{
233. cout << "> " << kata << endl;
234. }
235. **return**;
236. }
237. }
238. **void** compile(){
239. string inputan;
240. **bool** selesai = **false**;
242. **while**(selesai == **false**){
243. cout <<"# ";
244. getline(cin,inputan);
245. state(inputan, &selesai);
246. }
247. }
248. **void** null(**int** matrix[19][128]){
249. **for**(**int** i = 0 ; i < 19 ; i++){
250. **for**(**int** j = 0 ; j < 128 ; j++){
251. matrix[i][j] = -1;
252. }
253. }
254. }
255. **int** main(){
256. null(matrix);
257. setNull();
258. compile();
260. **return** 0;
261. }

## 3.2 Output

## 3.3 Rancangan Mesin Turing

Mesin Turing adalah model yang sangat sederhana dari komputer. Secara esensial, mesin Turing adalah sebuah finite automaton yang miliki sebuah tape tunggal dengan panjang tak terhingga yang dapat membaca dan menulis data.

Mesin Turing menggunakan notasi seperti ID-ID pada PDA untuk menyatakan konfigurasi dari komputasinya.

Mesin terdiri dari sebuah finite control, yang dapat berada dalam sebuah himpunan berhingga dari state. Terdapat sebuah tape yang dibagi ke dalam kotak-kotak atau sel-sel. Setiap sel dapat menampung sebuah dari sejumlah berhingga dari simbol.

Pada awalnya, input yang merupakan string dari simbol dengan panjang berhingga dipilih dari input alphabet, ditempatkan pada tape.

• Sel-sel tape yang lain, perluasan secara infinite ke kiri dan ke kanan, pada awalnya menampung simbol khusus yang dinamakan blank.

• Blank bukan sebuah input symbol, dan mungkin terdapat simbol tape yang lain disamping input symbol dan blank.

• Terdapat sebuah tape head yang selalu ditempatkan pada salah satu dari sel-sel tape.

• Mesin turing dikatakan men-scan sel tersebut. Pada awalnya, tape head berada pada sel paling kiri yang menampung input.

Sebuah pergerakan mesin Turing adalah sebuah fungsi dari state dari finite control dan tape symbol yang di-scan.

Dalam satu pergerakan, mesin Turing akan:

 Merubah state. Next state dapat sama dengan current state. Menulis sebuah tape symbol dalam sel yang di-scan. Tape symbol ini mengganti symbol apapun yang ada dalam sel tersebut. Secara opsional, simbol yang dituliskan dapat sama dengan simbol yang sekarang ada dalam tape. Memindahkan tape head ke kiri atau ke kanan.

Mesin Turing dijelaskan oleh 7-tuple:

M = (Q, S, G, d, q0, B, F)

Komponen-komponennya adalah:

• Q: Himpunan berhingga dari state dari finite control.

•S: himpunan berhingga dari simbol-simbol input.

• G: Himpunan dari tape symbol. S merupakan subset dari G.

• d: Fungsi transisi. Argumen d(q, X) adalah sebuah state q dan sebuah tape symbol X. Nilai dari d(q, X), jika nilai tersebut didefinisikan, adalah triple (p, Y, D), dimana:

•p adalah next state dalam Q

• Y adalah simbol, dalam G, ditulis dalam sel yang sedang di-scan, menggantikan simbol apapun yang ada dalam sel tersebut.

• D adalah arah, berupa L atau R, berturut-turut menyatakan left atau right, dan menyatakan arah dimana head bergerak.

• q0: start state, sebuah anggota dari Q, dimana pada saat awal finite control ditemukan.

• B: simbol blank. Simbol ini ada dalam G tapi tidak dalam S, yaitu B bukan sebuah simbol input.

• F: himpunan dari final state, subset dari Q.

# **BAB IV KESIMPULAN DAN SARAN**

## 4.1 Kesimpulan

Adapun kesimpulan dari laporan ini adalah sebagai berikut

* + - 1. Interpreter ini hanya interpreter sederhana yang memenuhi spesifikasi khusus.
      2. Dalam perancangan program memanfaatkan teori otomata jenis *Context Free*.
      3. Dalam pembuatan program menggunakan pemrograman dengan bahasa C++.

## 4.2 Saran

Adapun saran dari laporan ini adalah sebagai berikut

Bisa membuat program interpreter dengan memanfaatkan teori otomata dengan jenis lain.

Bisa membuat program interpreter dengan memanfaatkan pemrograman berorientasi objek dengan bahasa java.