PENERJEMAH BAHASA ALAMI DALAM BAHASA INDONESIA KE *SOURCE CODE* DALAM BAHASA *PASCAL*

Mohammad Kohar[1], Ken Kinanti Purnamasari[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung 40132
Email : kohar@email.unikom.ac.id[1], ken.kinanti@email.unikom.ac.id[2]

ABSTRAK

Source code dalam dunia komputer merupakan kumpulan dari perintah untuk menyelesaikan suatu masalah yang ditulis dalam suatu bahasa pemrograman. Namun bahasa pemrograman biasanya sulit untuk dipahami, dikarenakan struktur bahasanya yang kaku dan tidak natural. Salah satu cara yang dapat dilakukan agar pemrogram tidak perlu memahami struktur suatu bahasa pemrograman adalah dengan translasi dari bahasa alami ke bahasa pemrograman. Oleh karena itu, penelitian ini akan melakukan translasi dari bahasa alami dalam bahasa Indonesia ke *source code* dalam bahasa *Pascal*. Metode yang digunakan pada penelitian ini adalah metode *rule-based*. Proses pada sistem memiliki tiga tahap utama, yaitu *preprocessing* (*case folding* dan *filtering*), analisis (*scanning* dan *parsing*), dan translasi (pembangkitan kode). Tahap *preprocessing* dilakukan agar teks masukan bersih dari karakter yang tidak dibutuhkan, kemudian tahap analisis merupakan proses memastikan teks masukan sesuai dengan aturan penulisan yang dibuat, kemudian masuk tahap translasi dari bahasa Indonesia ke bahasa *Pascal*. Penelitian ini dapat *mentranslasi* perintah runtunan yang meliputi pembuatan variabel, pemanggilan fungsi readln, writeln, dan operasi aritmetika dasar. Berdasarkan hasil pengujian terhadap 100 teks perintah bahasa Indonesia yang mengandung 8 kombinasi perintah menghasilkan nilai akurasi sekitar 98%. *Grammar* bahasa Indonesia dan bahasa *Pascal* perlu disesuaikan kembali agar dapat menangani perintah yang lebih kompleks.

Kata kunci : Translasi, Kode Sumber, Bahasa *Pascal*, Pemrosesan Bahasa Alami, Bahasa Pemrograman

1. PENDAHULUAN

Source code atau kode sumber dalam dunia komputer merupakan kumpulan-kumpulan perintah untuk menyelesaikan masalah atau algoritma yang ditulis dalam bahasa yang dimengerti oleh komputer atau biasa disebut bahasa pemrograman [1]. *Source code* memiliki aturan penulisan yang sudah ditentukan oleh bahasa pemrograman tertentu yang digunakan. Ketika pemrogram akan membuat suatu pernyataan atau perintah untuk melakukan sebuah proses, maka harus mengikuti aturan penulisan yang ditentukan oleh suatu bahasa pemrograman. Namun, bahasa pemrograman sulit untuk dipahami, karena struktur penulisannya terkesan kaku atau tidak natural. Salah satu cara agar dapat membuat program tanpa harus memahami struktur bahasa pemrograman adalah dengan translasi dari bahasa alami ke bahasa pemrograman.

Beberapa penelitian sudah dilakukan untuk translasi dari bahasa Inggris ke *source code* dalam bahasa *Python* [2] dan C [3], dan dari bahasa Indonesia ke *source code* dalam bahasa C++ [4]. Penelitian yang dilakukan oleh Satu dan Avinash [2] sudah dapat menerjemahkan dari bahasa alami dalam bahasa Inggris ke *source code*. Pada penelitian tersebut [2] dapat menangani kasus runtunan seperti memasukkan nilai, penjumlahan, dan menampilkan nilai. Penelitian tersebut [2] tidak melakukan pengujian akurasi, sehingga nilai akurasi yang didapat belum dapat diketahui. Kemudian pada penelitian yang dilakukan oleh Nadkarni, Panchmatia, Karwa, dan Kurhade [3] dan penelitian yang dilakukan oleh Dirgahayu, Huda, Zukhri, dan Ratnasari [4], dapat menangani kasus runtunan, percabangan, dan pengulangan. Hanya saja pada penelitian tersebut [3] [4] masih menggunakan teks masukan dengan format *pseudocode* dalam bahasa Inggris [3] dan bahasa Indonesia [4]. Sampai saat ini, peneliti belum menemukan penelitian pada kasus translasi dari bahasa alami dalam bahasa Indonesia ke *source code*.

Berdasarkan uraian di atas, penelitian ini akan membangun sistem penerjemah dari bahasa alami dalam bahasa Indonesia ke *source code* dalam bahasa Pascal. Pada tahap proses translasi menggunakan metode *rule-based* yang ditentukan berdasarkan kasus yang diangkat.

2. LANDASAN TEORI

Landasan teori merupakan teori-teori yang menjadi acuan dalam penelitian ini.

2.1. Pemrosesan Bahasa Alami

Pemrosesan bahasa alami merupakan salah satu cabang dari kecerdasan buatan. Pemrosesan bahasa alami mengkaji komunikasi antara manusia dengan

komputer dengan perantara bahasa alami yang dimiliki manusia. Dalam melakukan hal tersebut, bahasa alami yang dikirimkan ke komputer harus diproses terlebih dahulu agar dimengerti oleh komputer [5]. Salah satu tantangan dalam pemrosesan bahasa alami adalah pemilihan arti dari sebuah kata yang memiliki makna lebih dari satu, misalnya kata 'bisa', kata 'bisa' dapat berarti 'racun' dan dapat juga berarti 'dapat' sesuai dengan bentuk kalimatnya [6]. Penelitian yang dapat menangani kasus kata yang memiliki lebih dari satu makna adalah dengan cara *Part of Speech Tagger*, seperti yang dilakukan oleh Purnamasari dan Suwardi [7]. Beberapa area penelitian dalam bidang pemrosesan bahasa alami adalah *question answering sistem, summarization, speech recognition, document classification, dan machine translation* [5]. *Machine translation* merupakan penelitian yang berfokus agar komputer memahami bahasa alami manusia yang dimasukkan dan menerjemahkannya ke bahasa lain [5].

2.2. Bahasa Pemrograman *Pascal*

Bahasa *Pascal* merupakan bahasa pemrograman yang dikembangkan oleh Niklaus Wirth sekitar tahun 1970 [8]. Nama Pascal diambil dari seorang ahli matematika yang bernama Blaise Pascal. Bahasa Pascal ini awalnya dikembangkan untuk pengajaran bahasa pemrograman.

2.3. *Scanning*

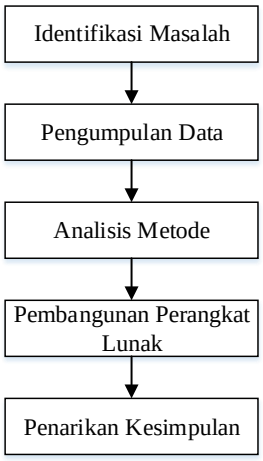
Scanning merupakan tahap pemecahan teks masukan menjadi token-token berdasarkan kelasnya [9]. Kemudian token-token hasil *scanning* akan menjadi data masukan pada tahap *Parsing*.

2.4. *Parsing*

Parsing merupakan tahap analisis sintaksis dengan melakukan pengecekan urutan kemunculan *string* atau token berdasarkan *grammar* yang sudah ditentukan. *Grammar* sendiri adalah kumpulan dari simbol non terminal, simbol terminal, dan simbol awal yang dibatasi oleh aturan produksi [9]. Sederhananya *parsing* merupakan proses pengecekan teks masukan apakah sudah sesuai dengan aturan bahasa yang digunakan atau belum.

3. METODE PENELITIAN

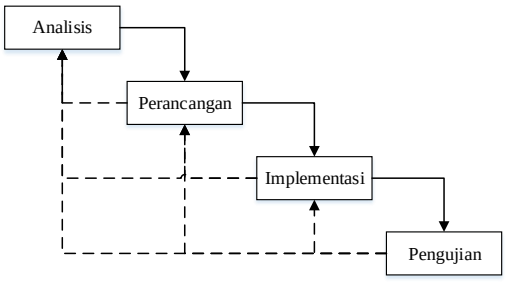
Pada penelitian ini, menggunakan metode penelitian deskriptif. Karena pada penelitian ini fakta dan karakteristik objek digambarkan secara sistematis [10]. Alur penelitian pada penelitian ini mencakup lima tahap, yaitu identifikasi masalah, pengumpulan data, analisis metode, pembangunan perangkat lunak, dan penarikan kesimpulan. Alur penelitian dapat dilihat pada Gambar 1.



Gambar 1 Alur Penelitian

Penjelasan dari alur penelitian pada Gambar xx adalah sebagai berikut.

a. Identifikasi Masalah
   Tahap identifikasi masalah merupakan proses pengamatan terhadap penelitian yang sudah pernah dilakukan sebelumnya untuk menentukan kebutuhan dan tujuan sistem yang akan dicapai.
b. Pengumpulan Data
   Pada penelitian ini, pengumpulan data yang dilakukan adalah studi literator baik secara tercetak maupun elektronik.
c. Analisis Metode
   Pada tahap analisis metode, metode yang digunakan dalam penelitian ini akan dianalisis, mulai dari tahap *preprocessing*, proses analisis, dan proses translasi.
d. Pembangunan Perangkat Lunak
   Pada penelitian ini, pembangunan perangkat lunak yang digunakan adalah model *waterfall*. Model *waterfall* merupakan metode pembangunan perangkat lunak yang bersifat sekuensial dalam tiap prosesnya [11] [12]. Alur dari model *waterfall* dapat dilihat pada Gambar 2.



Gambar 2 Model *Waterfall* [12]

e. Penarikan Kesimpulan
   Pada tahap penarikan kesimpulan merupakan penjelasan hasil dari penelitian yang sudah dilakukan.

4. HASIL DAN PEMBAHASAN

Pada bab ini akan membahas mengenai penelitian yang dilakukan dan hasil yang didapat dari penelitian ini.

4.1. Analisis Masalah

Source code merupakan kumpulan dari perintah untuk menyelesaikan suatu masalah yang ditulis mengikuti bahasa pemrograman yang digunakan. *Source code* format penulisan *source code* tidaklah alami sehingga sulit untuk dipelajari strukturnya. Oleh karena itu diperlukannya sebuah sistem yang dapat menerjemahkan bahasa alami dalam bahasa Indonesia ke *source code* dalam bahasa *Pascal*. Berikut adalah contoh translasi dari bahasa Indonesia ke bahasa *Pascal*.

Tabel 1 Contoh Hasil Translasi

Bahasa Indonesia	Bahasa *Pascal*

Buat program penjumlahan. Kemudian buat variabel a dengan tipe data integer. Tambahkan 15 dengan 2 masukan ke a. Tampilkan nilai a.	program penjumlahan;
	var
	a : integer;
	begin
	a := 15 + 2;
	writeln(a);
	end.

Buat aplikasi hitungluas. Buat variabel sisi dan luas dengan tipe data integer. Baca nilai sisi. Kalikan sisi dengan sisi kemudian masukkan hasilnya ke luas. Tampilkan nilai luas.	program hitungluas ; var sisi , luas : integer ; begin readln (sisi) ; luas := sisi * sisi ; writeln (luas) ; end.

4.2. Analisis Data Masukan

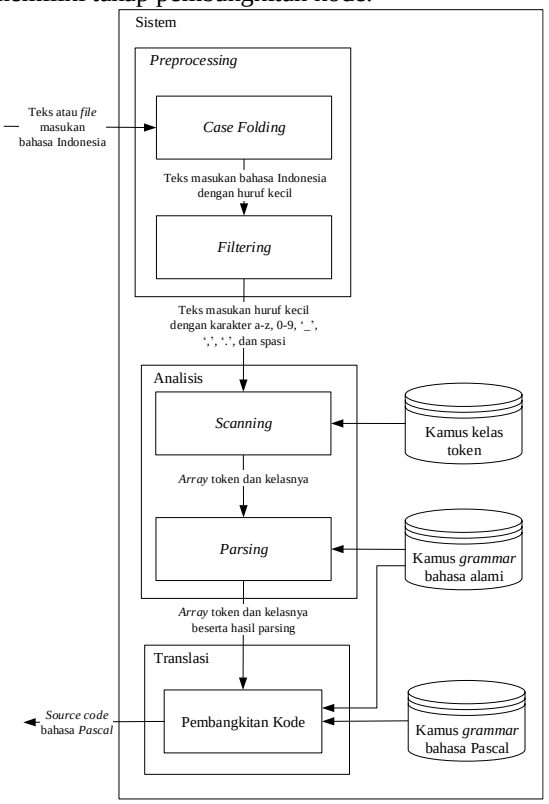
Pada penelitian ini data masukan berupa teks bahasa Indonesia yang berisi kalimat-kalimat yang beruntun untuk menyelesaikan sebuah masalah. Contoh dari data masukan pada penelitian ini dapat dilihat pada Tabel 2.

Tabel 2 Contoh Teks Masukan

No.	Teks Masukan
1	Buat aplikasi inputoutput. Buat variabel x dengan tipe data integer. Baca nilai x. Kemudian tampilkan nilai x.
2	Buat aplikasi hitungvolume. Kemudian buat variabel panjang, lebar, tinggi, dan volume dengan tipe data integer. Baca nilai panjang, lebar, tinggi. Kemudian panjang dikali lebar dikali tinggi masukkan hasilnya ke volume. Kemudian tampilkan nilai volume!.
3	Buat aplikasi hitungluas. Buat variabel sisi dan luas dengan tipe data integer. Baca nilai sisi. Kalikan sisi dengan sisi kemudian masukkan hasilnya ke luas. Tampilkan nilai luas.

4.3. Gambaran Umum Sistem

Pada penelitian ini, sistem yang dibangun akan dapat menerjemahkan bahasa Indonesia ke *soirce code* dalam bahasa *Pascal*. Sistem yang dibangun memiliki tiga proses utama, yaitu *preprocessing*, proses analisis, dan yang terakhir adalah proses translasi. Pada *preprocessing* memiliki tahapan *case folding* dan *filtering*, proses analisis memiliki tahapan *scanning* dan *parsing*, dan proses translasi memiliki tahap pembangkitan kode.



Gambar 3 Blok Diagram Sistem Keseluruhan

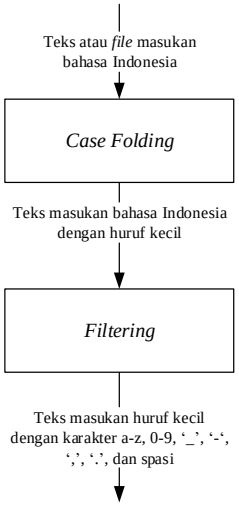
Proses akan dijalankan ketika pengguna memasukkan teks perintah pembuatan program dalam bahasa Indonesia.

Tabel 3 Contoh Teks Masukan

Teks Masukan
Buat aplikasi tampil_string. Tampilkan hallo world!.

4.4. *Preprocessing*

Preprocessing merupakan tahap awal untuk mempersiapkan teks masukan bahasa Indonesia agar siap digunakan pada proses analisis. *Preprocessing* memiliki dua tahap, yaitu *case folding* dan *filtering*.



Gambar 4 Blok Diagram *Preprocessing*

a. *Case Folding*
Pada penelitian ini *case folding* digunakan untuk menyeragamkan huruf menjadi *lower case* atau huruf kecil untuk memudahkan pada proses analisis.

Tabel 4 Contoh Hasil *Case Folding*

Sebelum
Buat aplikasi tampil_string. **T**ampilkan hallo world!.
Sesudah
buat aplikasi tampil_string. **t**ampilkan hallo world!.

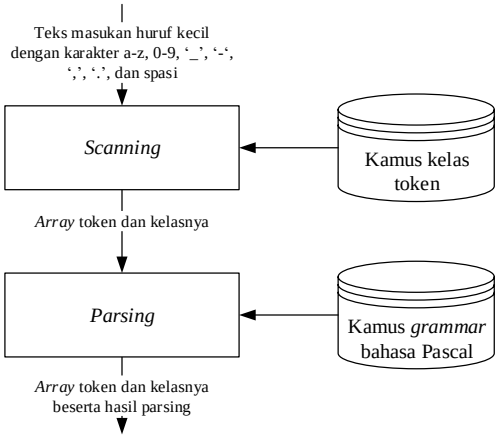
b. *Filtering*
Tahap *filering* merupakan tahap pemilahan karakter yang dianggap diperlukan dalam penelitian ini. Karakter-karakter yang tidak diperlukan akan dihilangkan atau diganti. Pada penelitian ini karakter yang diperbolehkan masuk tahap translasi adalah karakter a-z, 0-9, '_', '-', koma (','), titik ('.'), dan spasi.

Tabel 5 Contoh Hasil *Filtering*

Sebelum
buat aplikasi tampil_string. tampilkan hallo world**!**.
Sesudah
buat aplikasi tampil_string. tampilkan hallo world.

4.5. Proses Analisis

Proses analisis merupakan tahap menganalisis data masukan apakah sesuai dengan aturan atau tidak. Proses analisis memiliki dua tahap, yaitu *scanning* dan *parsing*.



Gambar 5 Blok Diagram Proses Analisis

a. *Scanning*
Scanning merupakan tahap mengubah teks masukan hasil *preprocessing* menjadi token-token dan kelasnya. Kemudian token-token hasil *scanning* akan digunakan pada tahap parsing dan proses translasi.

Tabel 6 Daftar Token dan Kelasnya

Token	**Kelas Token**
'tambah', 'ditambah', 'tambahkan', 'ditambahkan', 'kurang', 'dikurang', 'kurangkan', 'dikurangkan', 'kurangi', 'dikurangi', 'kali', 'dikali', 'kalikan', 'dikalikan', 'bagi', 'dibagi', 'bagikan', 'dibagikan'.	Arithmetic Operator
'program', 'aplikasi', 'variabel', 'peubah', 'tipe', 'tipenya', 'integer', 'string', 'bulat', 'pecahan', 'masuk', 'masukkan', 'dimasukkan', 'isi', 'isikan', 'diisi', 'diisikan', 'tampil', 'tampilkan', 'baca'.	Keyword
'buat', 'buatkan', 'buatlah', 'kemudian', 'lalu', 'dan', 'dengan', 'ke', 'data', 'datanya', 'nilai', 'nilainya', 'hasil', 'hasilnya', 'bilangan'.	Additional Token
[a..z, 0..9, '_']	IdentApp
[a..z, 0..9, '_']	IdentVar
[0..9, ',' , '-']	Number
[a..z, 0..9, '_']	String
',', '.'	Delimiter

Contoh hasil *scanning* dari data hasil *filtering* yang ada pada Tabel 5 dapat dilihat pada Tabel 7.

Tabel 7 Contoh Hasil *Scanning*

Sebelum
buat aplikasi tampil_string. tampilkan hallo world.

Sesudah	
Token	**Kelas**
buat	AdditionalToken
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter

b. *Parsing*

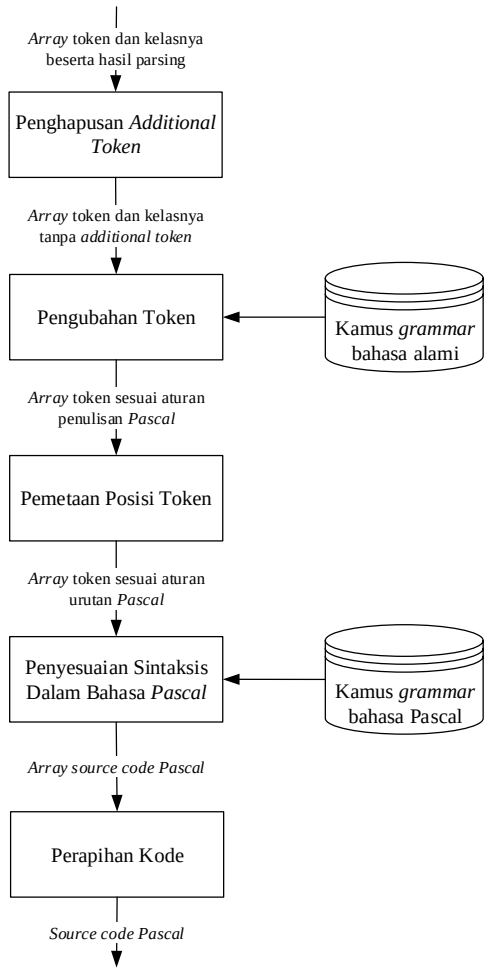
Tahap *parsing* merupakan tahap pengecekan urutan kemunculan teks masukan bahasa Indonesia hasil proses *scanning*. Tujuan dari proses parsing adalah memastikan urutan kemunculan token sesuai dengan aturan yang telah dibuat, sehingga jika terdapat teks masukan yang tidak sesuai dengan aturan atau *grammar* proses translasi tidak akan dijalankan. Pada contoh kasus pada penelitian ini status *parsing* adalah diterima, karena semua token berhasil diturunkan sesuai *grammar*, penurunan token dapat dilihat pada Tabel 8.

Tabel 8 Contoh Proses Penurunan Token

<START> → <PROGRAM_DECL <DELIMITER BLOCK>
<COMMAND_WORD> <PROGRAM_KEYWORD> <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat <PROGRAM_KEYWORD> <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat aplikasi <PROGRAM_IDENT> <DELIMITER> <BLOCK>
buat aplikasi tampil_string <DELIMITER> <BLOCK>
buat aplikasi tampil_string . <BLOCK>
buat aplikasi tampil_string . <STATEMENT>_<BLOCK>
buat aplikasi tampil_string . <STATEMENT_SQUENCE> <DELIMITER>
buat aplikasi tampil_string . <STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <IO_STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <OUTPUT_STATEMENT> <DELIMITER>
buat aplikasi tampil_string . <KEYWORD_OUTPUT> <EKSPRESI_SEQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI_SEQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI> <DELIMITER>
buat aplikasi tampil_string . tampilkan <EKSPRESI_1> <DELIMITER>
buat aplikasi tampil_string . tampilkan <FACTOR_SQUENCE> <DELIMITER>
buat aplikasi tampil_string . tampilkan <FACTOR> <DELIMITER>
buat aplikasi tampil_string . tampilkan <STRING> <DELIMITER>
buat aplikasi tampil_string . tampilkan hallo world <DELIMITER>
buat aplikasi tampil_string . tampilkan hallo world .

4.6. Proses Translasi

Proses translasi merupakan proses penerjemahan teks bahasa Indonesia ke *source code* bahasa *Pascal*. Proses translasi ini akan dijalankan ketika status parsing pada tahap analisis berstatus 'diterima'. Proses translasi memiliki satu tahap, yaitu pembangkitan kode.



Gambar 6 Blok Diagram Pembangkitan Kode

a. Penghapusan *Additional Token*

Penghapusan *additional token* merupakan tahap penghapusan token yang memiliki kelas 'AdditionalToken', karena pada penelitian ini token tersebut dianggap tidak diperlukan.

Tabel 9 Contoh Hasil Penghapusan *Additional Token*

Sebelum	
Token	**Kelas**
buat	AdditionalToken
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter
Sesudah	
Token	**Kelas**
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter

b. Pengubahan Token
Pada tahap pengubahan token, token yang memiliki kelas 'Keyword', 'ArithmeticOperator' dan 'Number' akan diubah sesuai dengan aturan dalam bahasa *Pascal*. Pengubahan ini memanfaatkan *Grammar* bahasa Indonesia yang sudah dibuat dan *Grammar* bahasa *Pascal*.

Tabel 10 Contoh Hasil Pengubahan Token

Sebelum	
Token	**Kelas**
aplikasi	Keyword
tampil_string	IdentApp
.	Delimiter
tampilkan	Keyword
hallo world	String
.	Delimiter
Sesudah	
program	Keyword
tampil_string	IdentApp
.	Delimiter
writeln	Keyword
hello world	String
.	Delimiter

c. Pemetaan Posisi Token
Pemetaan posisi token merupakan tahap penyesuaian urutan kemunculan token sesuai dengan aturan bahasa *Pascal*. Pemetaan token pada penelitian ini dilakukan untuk perintah *assignment*.

d. Penyesuaian Sintaksis Dalam Bahasa *Pascal*
Pada tahap ini mirip dengan proses *parsing*, hanya saja bukan bertujuan untuk mengecek urutan kemunculan token, melainkan untuk memasukkan token-token hasil pemetaan token ke *grammar* bahasa *Pascal*, karena ada beberapa kode yang tidak ada dalam penulisan bahasa alami, misalnya 'begin', 'end', dan sebagainya.

Tabel 11 Contoh Hasil Penyesuaian Sintaksis Dalam Bahasa *Pascal*

Sebelum	Sesudah
program	program
tampil_string	tampil_string
.	;
writeln	begin
hello world	writeln
.	(
	'
	hello world
	'
	)
	;
	end.

e. Perapihan Kode
Pada tahap terakhir, token yang sudah disisipi kode yang dibutuhkan dalam bahasa *Pascal* akan dirapikan agar mudah dipahami.

Tabel 12 Contoh Hasil Perapihan Kode

Sebelum	Sesudah
program	program tampil_string ; begin writeln ('hallo world') ; end.
tampil_string	
;	
begin	
writeln	
(	
'	
hello world	
'	
)	
;	
end.	

4.7. Analisis Hasil Pengujian

Pengujian akurasi dilakukan dengan cara membandingkan hasil translasi yang didapat dari sistem dengan hasil yang diharapkan. Pengujian dilakukan pada perintah pembuatan variabel, pemanggilan fungsi *readln*, pemanggilan fungsi *writeln*, dan operasi aritmetika dasar. Pengujian dilakukan pada 100 data uji yang dibagi ke dalam 8 kombinasi perintah. Hasil dari pengujian translasi dari bahasa Indonesia ke *source code* dalam bahasa *Pascal* dapat dilihat pada Tabel 13.

Tabel 13 Hasil Pengujian Akurasi

Kombinasi Perintah	Jumlah Data Uji	Hasil Translasi	
		Benar	Salah
Tanpa variabel, 1 statement	10	9	1
Tanpa variabel,	10	9	1

lebih dari 1 statement			
1 variabel, 1 statement	10	10	0
1 variabel, lebih dari satu statement	10	10	0
Lebih dari 1 variabel, lebih dari 1 statement	10	10	0
1 operator aritmetika	15	15	0
2 operator aritmetika	18	18	0
Lebih dari 2 operator aritmetika	17	17	0
Total	100	98	2

Nilai akurasi yang didapat pada penelitian ini adalah 98%. Kesalahan translasi disebabkan pada tahap *scanning* sistem tidak bisa membedakan bilangan pecahan desimal dengan dua angka yang dipisahkan dengan koma. Contoh kalimat yang salah deteksi dapat dilihat pada Tabel 14.

Tabel 14 Kesalahan Hasil *Scanning*

Teks Masukan	Hasil Diharapkan	Hasil Dari Sistem
tampilkan 1, 2, dan 3.	tampilkan	tampilkan
	1	**1,2**
	,	,
	2	dan
	,	3
	dan	.
	3	
	.	

Pada penelitian ini terdapat beberapa kasus yang belum dapat ditangani, dan dapat dijadikan acuan pada penelitian selanjutnya. Kasus yang belum dapat ditangani adalah sebagai berikut.

a. Sistem belum dapat membedakan bilangan pecahan desimal dengan dua buah angka yang dipisahkan dengan koma, seperti pada Tabel 14.
b. Kalimat masukan yang masih terbatas dikarenakan *grammar* bahasa Indonesia pada penelitian ini masih sederhana. Contoh kalimat yang sudah dapat ditangani dan belum dapat dilihat pada Tabel 15.

Tabel 15 Contoh Perintah *Assignment* Yang Ditangani Dan Belum Ditangani

Contoh Kalimat	Status
Tambahkan 12 dengan 1 masukkan hasilnya ke hsl.	
12 ditambah 1 masukkan hasilnya ke hsl.	Ditangani
Hsl diisi dengan hasil penjumlahan 12 dan 1.	Belum Ditangani
Jumlahkan 12, 1, dan 2 masukkan hasilnya ke hsl.	Belum Ditangani

c. Sistem belum dapat menangani kasus percabangan dan perulangan, misalnya pada Tabel 16.

Tabel 16 Contoh Perintah Percabangan Dan Perulangan

Perintah Percabangan
Jika indeks tidak sama dengan e, maka tampilkan lulus, namun jika tidak tampilkan tidak lulus.
Perintah Perulangan
Selama i kurang dari 10, maka tampilkan i.

d. Sistem belum dapat menganalisis data masukan secara semantik, seperti pada Tabel 17.

Tabel 17 Contoh Perintah Yang Salah Secara Semantik

Teks Masukan	Keterangan
Buat aplikasi uji_string. Buat variabel nama_depan dan nama_belakang dengan tipe data string. Baca nama_depan dan nama_belakang. Tampilkan hasil nama_depan dikali nama_belakang.	Terdapat perintah yang mengalikan dua buah variabel bertipe data *string*.
Buat program uji_coba. Buat variabel a dengan tipe data string. Kemudian 12 dimasukkan ke a.	Terdapat perintah memasukkan nilai *integer* ke dalam variabel *string*.

e. Sistem belum dapat menangani kesalahan eja pada teks masukan, misalnya pada Tabel 18.

Tabel 18 Contoh Kesalahan Eja Pada Teks Masukan

Masukan	Keterangan
Buat aplikasi uji_string. Tampikan hallo world.	Terdapat kesalahan eja pada kata ‘tampikan’, seharusnya ‘tampilkan’.

5. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem dapat menerjemahkan bahasa alami dalam bahasa Indonesia ke *source code* dalam bahasa *Pascal*. Penelitian ini mendapat nilai akurasi sebesar 98%.

Terdapat beberapa kasus yang belum dapat ditangani pada penelitian ini, oleh karena itu penelitian ini dapat dikembangkan kembali dikemudian hari. Terdapat saran yang dapat diterapkan pada penelitian selanjutnya, yaitu sebagai berikut.

a. Menambahkan aturan yang dapat membedakan bilangan pecahan dengan dua angka yang dipisahkan dengan koma seperti pada Tabel 4,14.
b. Menambahkan kompleksitas *grammar* bahasa Indonesia dan bahasa *Pascal* agar dapat mendeteksi kalimat yang lebih beragam.
c. Menambahkan fitur translasi untuk struktur dasar percabangan dan pengulangan.
d. Menambahkan fitur analisis semantik pada tahap analisis teks masukan bahasa Indonesia.
e. Menambahkan fitur koreksi kata yang salah eja.

DAFTAR PUSTAKA

[1] R. Munir, Algoritma dan Pemrograman, Bandung: Informatika, 2011.

[2] D. Satu dan A. Avinash, "Unrestricted Natural Language Implementation in Programming," *International Research Journal of Engineering and Technology*, vol. 03, no. 10, pp. 470-476, 2016.

[3] S. Nadkarni, P. Panchmatia, T. Kaewa dan S. Kurhade, "Semi Natural Language Algorthm to Programming Language Interpreteur," *International Conference on Advances in Human Machine Interaction*, 2016.

[4] T. Dirgahayu, S. N. Huda, Z. Zukhri dan C. I. Ratnasari, "Automatic Translation from Pseudocode to Source Code: A Conceptual-Metamodel Approach," *2017 IEEE International Conference on Cybernetics and Computational Intelligence, CyberneticsCOM 2017 - Proceedings*, pp. 122-128, 2018.

[5] W. Budiharto dan D. Suhartono, Artificial Intelligence, Yogyakarta: Andi Offset, 2014.

[6] M. Arhami, Konsep Kecerdasan Buatan, Yogyakarta: Andi Offset, 2006.

[7] K. K. Purnamasari dan I. S. Suwardi, "Rule-based Part of Speech Tagger for Indonesian Language," *IOP Conference Series: Materials Science and Engineerin*, vol. 407, pp. 1-4, 2018.

[8] E. Nugroho, Bahasa-Bahasa Pemrograman, Yogyakarta: Andi Offset, 1992.

[9] F. Utdirartotmo, Teknik Kompilasi, Yogyakarta: Graha Ilmu, 2005.

[10] M. Nazir, Metode Penelitian, Bogor: Ghalia Indonesia, 2014.

[11] R. S. Pressman, Software engineering : A Practitioner's Approach Edisi 7, New York: McGraw-Hill, 2010.

[12] Y. Bassil, "A Simulation Model for the Waterfall," *International Journal of Engineering & Technology*, vol. 2, no. 5, 2012.