# Bahasa Pemrograman Kinuy

*by* Tedy Setiadi

**Submission date:** 02-Nov-2020 05:55AM (UTC+0700)
**Submission ID:** 1433002246
**File name:** Perancangan_dan_Implementasi_Bahasa_Pemrograman_Kinuy.docx (1.14M)
**Word count:** 1661
**Character count:** 10674

# PERENCANGAN DN IMPLEMENTASI BAHASA PEMOGRAMAN KINUY (BAHASA PEMROGRAMAN PRIBUMI)

**Tedy Setiadi**[1], **Arie Musbandi**[2]

***ABSTRACT***:*This research has developed Kinuy language, which is a programing language that close an Indonesian language and granumatical. This programming language is based on GDL (Grammar Definition Language) using ProGrammar software and become implemented to an interpreter using Visual Basic 6.0 GDL ProGrammar is used for a recognizer or a lexical analysis, also Visual Basic 6.0. GDL user as semantic analysis. This Interpreter was developed using a Syntax Direct Difinition method recursively. The Kinuy Language expectation is for helping a beginner Indonesion Progeammer.*
*Keyword : Grammar Definition Language, Recognizer, Interpreter.*

## PENDAHULUAN

Manusia dapat melakukan interaksi secara efektif dengn mengguakan media bahasa. Bahasa memungkinkan penyampaian gagasan dan pemikiran, tanpa kedua hal itu komunikasi kan sulit terjadi. Dalam lingkungan pemrograman computer, bahasa pemrograman bertindak sebagai sarana komunikasi antara manusia dan permasalahannya dengan computer yang dipakai untuk membantu memperoleh pemacahan. Bahasa pemrograman mejembatani antara pemikiran manusia yang sering tidak terstruktur dengan kepastian yang diperlukan oleh computer untuk eksekusi. Suatu solusi untuk suatu masalah akan menjadi lebih mudah bila bahasa pemrograman lebih dekat dengan permasalahan tersebut.[6}. Oleh karena itu, bahasa harus memiliki konstruksi yang merefleksikan masalah dan independen dari Komputer yang dipergunakan. Komputer digital, disisi lain, hanya menerima dan memahami bahasa tingkat rendah mereka sendiri, terdiri dari deretan nol dan satu, yang sulit dipahami oleh manusia.

Ada banyak bahasa tingkat tinggi yang telah diciptakan manusia seperti Pascal, BASIC, C, C++, Java dan lain sebagainya. Mungkin untuk sebagian rekayasawan Indonesia yang sudah familiar dengan bhasa tersebut tidak akan sulit berkomunikasi dengan computer. Tetapi untuk sebagian rekayasawan Indonesia lain yang baru belajar berkomunikasi dengan computer akan merasa sulit untuk memahami bahas tersebut.
Ini dikarenakan tidk satu pun bahasa-bahas ini mirip dengan Bahasa Indonesia, sebagai contoh instruksi "*for*" yang digunakan untuk *loop-ing* atau perulangan akan sulit dimengerti oleh para pemula pemgrograman computer Indonesia.

Memanng ada sekelompok pemrograman Indonesia yang membuat bahasa pemrograman yang mirip dengn Bahasa Indonesia, seperti KILANG (BASIC Indonesia yang dibuat oleh Prof. Daly S. Naga dan beberapa pemrograman yang membuat bahas pemrograman "BATAK". Tetapi bahasa-bahasa tersebut sudah hulang jejaknya seiring dengan perkembagan bahasa pemrograman yang sudah maju. Bahasa pemrograman JAVA, BALI, MADURA haya nama saja, tetapi tidak stupun bahasa tersebut yang mirip dengan bahasa-bahasa yang digunakan di Indonesia [7].

---

[1] *Tedy Setiadi : Dosen Teknik Informatika Universitas Ahmad Dahlan Yogyakarta*
[2] *Arie Musbandi : Alumini Tekni Informatika Universitas Ahmad Dahlan Yogyakarta*

**PEMBAHASAN**

a. Pembuatan Spesifikasi Bahasa

Sebelum menentukan tata bahasa yang akan dibuat hal pertama yang harus dilakukan adalah menetukan spesifikasi bahas tersebut. Ini dibutuhkn sebagai patokan dalam pembuatan bahasa.
Adapun spesifikasi asa yag akan dibuat adalah sebagai berikut.

1. *Grammar* dengan pendekatan Bahasa Indonesia.
2. Pendeklarasian variabel, konstanta, fungsidan prosedur.
3. Menangani tipe data *integer, Flood, String* dan *Boolean*.
4. Sudah *terdapat* fungsi-fugsi bawaan sepertin min, max, avg, dan sqr.
5. *Selection statement if-then-else*.
6. *Looping statement for and while*
7. Dapat menangani *komentar* program.

b. Perancangan Tata Bahasa

Tata bahasa atau *grammar* adalah sistem matematis untuk mendeskripsikan bahasa. Dengan membentuk *grammar* proses persing akan menjadi lebh mudah. Dalam pembuatan diagram keadaan dan *grammar* menggunakan perangkat Bantu JFLAPBeta Versi 2.0 keunggulan perangkat ini adalah bisa mengkonversi diagram menjadi *grammar*.

Berdasarkan spesifikasi bahasa, Bahasa Kinuy sudah dapat megangani pemuatan fungsi prosedur, variabel dan konstanta. Artinya bagan program terdiri dari blok-blok fungsi dan fungsi utama. Dan uyang harus dipertimbangkan adalah *life scope* dari sebuah variabel. Pada bagian ini juga dilakukan penentuan bentuk sintak dari *statement*. Sebagai contoh bentuk intak *statement* "if"

Ditentukan sebagai berikut.

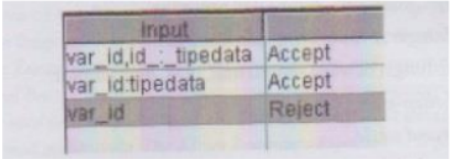
*JIKA <Expresi> MAKA*
*[STATEMENT]*
*MELAINKAN_JIKA <Expresi>*
*MELAINKAN*
*[STATEMENT]*
*AKHIR JIKA*

Diagram keadaan atau *state transition diagram* yang temasuk kedalam *Finite automata* adalah model matematika dengan memasukkan dan keluaran diskrit. Sistem dapat berada di salah satu dari sejumlah berhigga konfigurasi internal yang disebut *state*. *State* sistem merupakan serngakayan informasi yag berkaitan dengn masukan-masukan sebelumnya yang memutuskan perilaku sistempada masukkan beriktnya. Pembuatan diagram keadaan adalah cara termudah mebuat I*recognizer* dari bahasa yag kita buat. *Recognizer* membaca string yang kita masukkan dan memberi keluaran YA jika string tersebut termasuk dalam bahasa yang buat atau TIDAK jika string tidak termasuk kedlam bahasa.

Misalkan diagram transisi untuk deklarasi variabel adalah sebagai berikut.

Gambar 1. Diagram Transisi untuk deklarasi variabel

Hasil dari *recognizer* diagram transisi diatas diperlihatkan pada gambar dibawah ini.



| Input | |
|---|---|
| var_id,id_:_tipedata | Accept |
| var_id:tipedata | Accept |
| var_id | Reject |

Gambar 2. Hasil *recognizer* Deklarasi Variabel

Tata bahas (*grammmar)* bisa didefinisikan secara formal sebagai kumpulan darihumpunan-himpunan variabel, simbol-simbol terminal, simbol awal, yag dibatasi oleh aturan-aturan produksi oleh sebuah bahasa. Jadi bahasa berisi semua string yang dapat dihasilkan dari aturan-aturan *grammar*. Pada tahun 1959 seorang ahli bernama Noam Chomsky[2] melakukan penggolongan bahasa menjadi empat yang disebut dengan Hirarki *chomsky* yang ditujukan untuk bahasa alami. Penggolongan tersebut bisa dilihat pada tabel 1. Berikut ini.

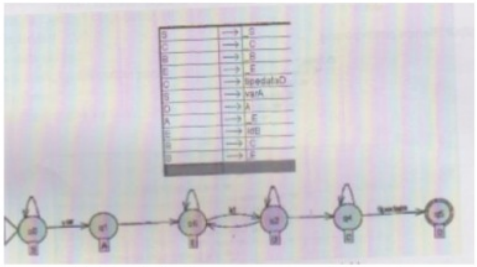
| **Bahasa** | **Mesin Otomata** | **Batasan Aturan Produksi** |
|---|---|---|
| Regular/Tipe 3 | Finite State Automta (FSA) meliputi *Deterministic Finite Automata* (DFA) dan *Nondterministic Finite Automata* (NFA) | α adalah sebuah simbol variabel<br>β maksimal memiliki satu simbol variabel yang bila ada terletak di posisi paling kanan |
| Bahasa Kontek/ *Context Free* /<br>Tipe 2 | *Push Down Automata* | α berupa sebuah simbol variabel |
| *Context Sensitive*/ Tipe 1 | *Linier Bounded Automata* | A \| ? \| β |
| *Unrestricted* /<br>*Phase Structure* /<br>*Natural Language* /<br>Tipe 0 | Mesin Turing | Tidak Ada Batasan |

Aturan produksi merupakan pusat dari tata bahasa, yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu *string* kebentuk lainnya, dn mellui aturan tertentu didefinisikan suatu bahasa yang berhi=ubungan dengan tata bahasa tersebut.

**a → b**

(a menghasilkan b)

Dimana a menyatakan simbol pada ruas kiri dan b menyatakan simbol-simbol pada ruas kanan dan bisa juga dinyatakan sebagai hasil produksi. Simbol-simbol tersebut bisa berupa simbol terminal atau simbol non-terminal/variabel. Simbol Variabel/non-terminal adalah simbol yang bisa ditunkan, sedngkan simbol terminal tidk bisa diturunkn lagi. Simbol terminal biasanya dinyatakan dengan huruf kecil misalnya 'a', 'b', 'c' sedangka simbol non-terminal diyatakan dengan hurf besar 'A', 'B', 'C'.



Gambar 3. Bentuk *Grammar* pendeklarasian variabel

c. Implementasi

GDL adalah sebuah bahasa yang merancang aturan-aturan bahasa. Sintak dari GDL mirip dengan BNF serta ditambah dengan penanganan kesalahan gramatikal.

Bentuk GDL dari pendeklarasian variabel adalah sebagai berikut.

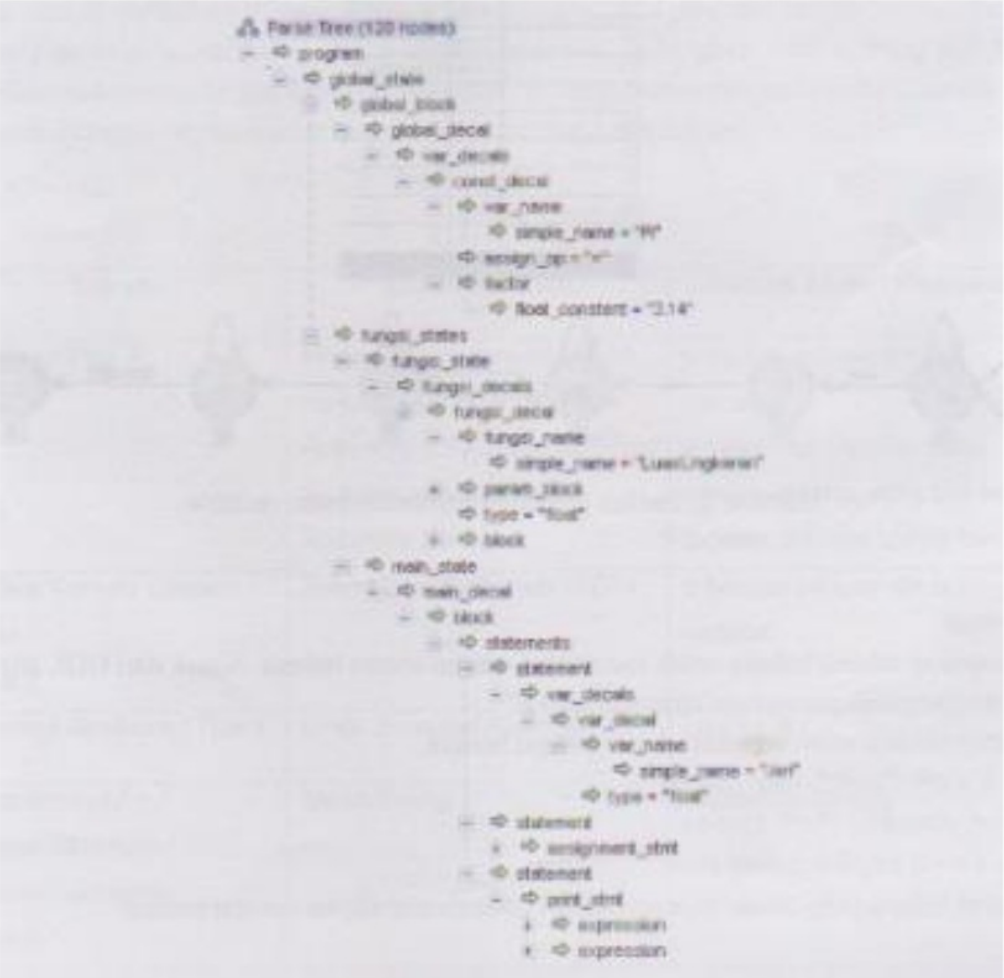
```
Var_decal ::= "VAR" var_name
[{"," var_name}] ":" type;
Var_name::= simple_name;
```

Hasil parsing untuk bahasa yang dibuat menggunakan ProGGrammar adalah sebagai berikut.

Input:

```
/*
      ** ------------------------------------------- **
      **    Cari Luas Lingkaran                      **
      ** ------------------------------------------- **
*/

global
      kon Pi = 3.14

fungsi LuasLingkaran (var r : float) : float
awal
      LuasLingkaran = 2 * Pi * sqr(r)
akhir

utama
awal
      var Jari : float

      Jari = input ("Masukkan Jari")
      cetak "Luas Lingkaran = "; LuasLingkaran(Jari)
akhir
```

Output: Berupa Pohon Parsing



Gambar 4. Pohon Parsing

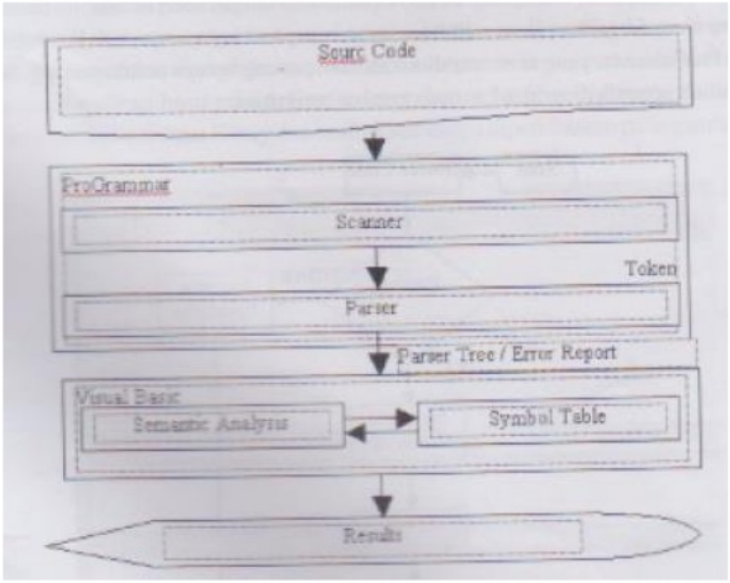
Interpreter yang dirancang hanya menentukan makna semnatik bahasa tersebut tanpa melakukan translasi kedalam bahasa lain. Interpreter ini dibuat dengan Microsoft Visual Basic 6.0 dan dengan bantuan ProGrammar untuk memudahkan implementasi bahasa. Fungsi dari ProGrammar selain sebagai *recognizer* bahasa juga memberikan *output* yang berupa pohon parsing. Sehingga akan mempercepat proses implementasi. Kelebihan lain yag bisa diperoleh dengan menggunakan ProGrammar adalah fungsi untuk penanganan kesalahan. Cara kerja ProGrammar dan Visual Basic diilustrasikan seperti gambar 5.

1. Perancangan Tabel Simbol

Tabel simbol merupakan seuah struktur data yang berfungsi menyimpan semua informasi selama prses interprestasi berlangsung. Semakin kompleks suatu bahasa maka penggunaaan struktur data padda tabel simbol juga semakin kompleks. Untuk bahasa KINUY menggunakan tabel simbol sederhana dengn memanfaatkan fasilitas Visual Basic yg disebut *User-Defined-Type* (UDT) yang memiliki komponen sebagai berikut.

*Symbol* : Nama dari simbol yang disimpan.
*Kind* : Jenis simbol.
*Type* : Tipe data simbol.

*nodeID* : Posisi simbol pada pohon parser.
*Value* : Nilai simbol.
*Parent* : Posisi blok simbol *life scope*.




Gambar 5. Bagan cara kerja ProGrammar dan Visual Basic pada pembuatan interpreter KINUY

2. Perancangan *Semantic Analysis*

Perancangan analisis semantik dilakukan dengan cara mentelusuri pohon parser dari *root* ingga ke *node* terbawah atau node tersebut merupakan simbol terminal sesuai dengan GDL yang telah dibuat. Pembuatan fungsi dan prosedur didasarkan pada GDL yang dihasilkan, non terminal ditelusuri hingga produksi menyatakan simbol terminal. Jadi GDL bisa dijadikan pengganti dari pengnalisis sintak.

Metode *syntax direct Definition* parser menerima input yg berupa token dan meghasilkan pohon parser yang kemudian ditentukan makna semantikn[10]dengan metode ini.[1]

contoh penerapan *syntx Direct definition* dapat dilihat pada tabel 2 dibawah ini.

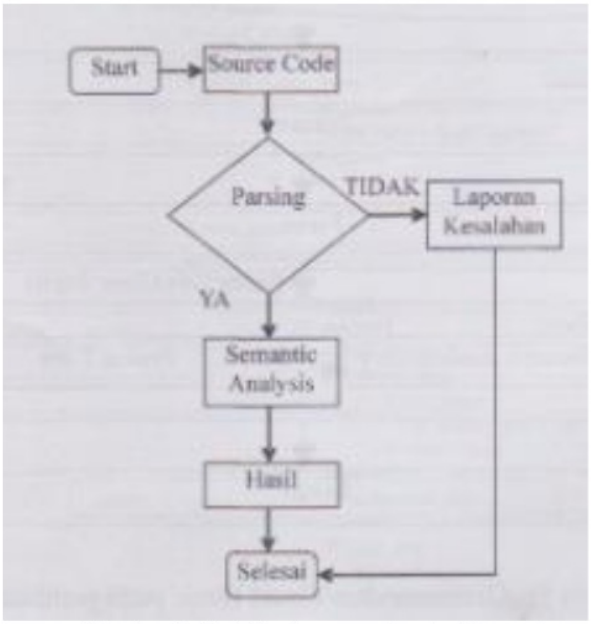
Tabel 2: *Syntax Direct Definition* pada kalkulator sederhana.

| **Aturan Produksi** | **Aturan Semantik** |
|---|---|
| L ? E | Print (L.EVAL) |
| E ? $E_1$ T | E.EVAL - $E_1$.EVAL + T.EVAL |
| E ? T | E.EVAL = T.EVAL |
| T ? $T_1$ * F | T.EVAL = $T_1$.EVAL * F.EVAL |
| T ? F | T.EVAL - F.EVAL |
| F ? E | F.EVAL - E.EVAL |
| F ? **digit** | F.EVAL -digit.LEXVAL |

b. Pengembangan Aplikasi

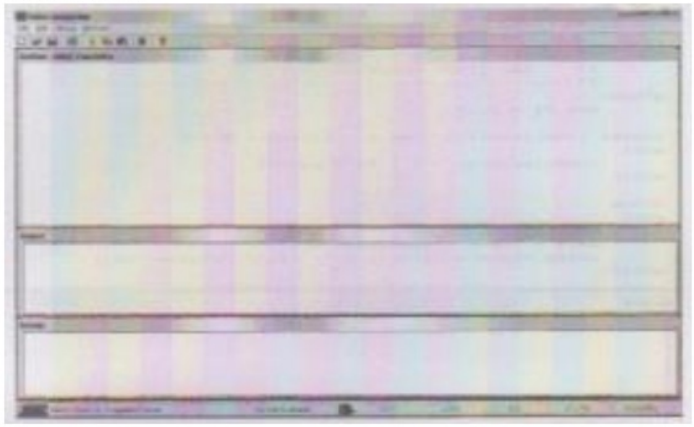
Interpreter yang akan dihasilkan akan membaca input berupa *string source code* kemudian di paring dengan menggunakan ProGarammar yang akan memberikan hasil parsing berupa pohon

parsing. Adapun skema global proses pengnalisis sematik dapa diperlihatkan pad gambar berikut ini.




Gambar 6. Global schame pembuatan interpreter

*Source code* yag di inputkan akan dianalisis oleh parser yang dilakukan ProGrammar apakah sesuai dengan *grammar* atau tata bahasa yang dibuat. Jika *source code* tersebut gagal diparsing yang artinya tidak sesuai dengan tata bahasa yang dibuat makan akan ditampilkan pesan kesalahan, apabila *source code* tersebut berhasil diparsing maka akan dilanjutkan pada bagian analisis semantik dan hasil pemaknaan semantik ditampilkan.

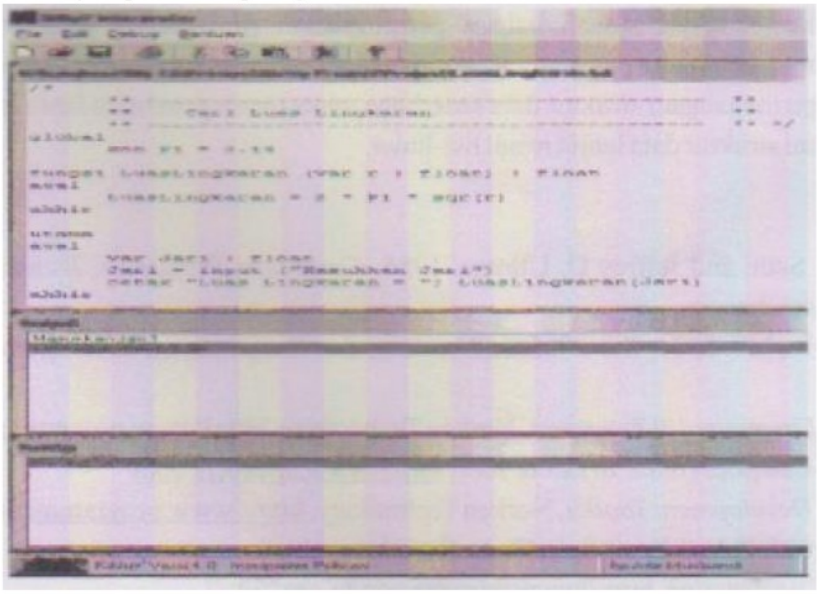


Gambar 7. Form Utama

c. Analisis Hasil

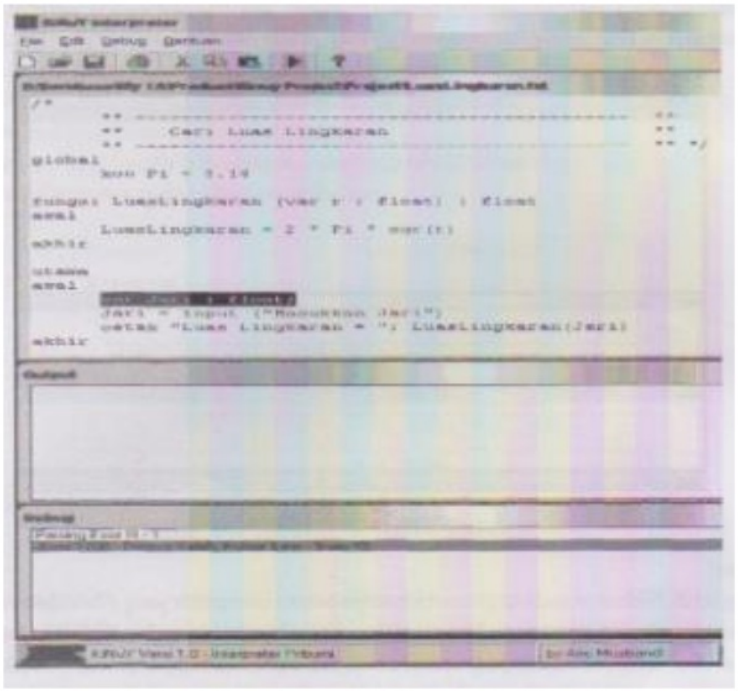
Seperti yang telah dijelaskan pada bagian sebelumnya bahwa interpreter yang dibuat hanya melakukan proses pemaknaan semantik dari input yang berupa *source code*. Jika hasil parsing berhasil yang artinya tidak ada kesalahan leksikal yang ditemukan pada *source code*, maka proses selanjutnya adalah melakukan pemaknaan semnatik dan menampilkan hasil pemaknaan tersebut

secara langsung (*direct*). Contoh *source code* yang memenuhi aturan produksi bahasa Kinuy dan hasil eksekusinya diperlihatkan pada gambar 8.



Gambar 8. Contoh *source code* yang sukses diparsing

Sedangkan jika *source code* tidak sesuai dengan tata bahasa yang ditetapkan maka proses pemaknaan semntik tidak dilanjutkan dan laporan kesalahan diberikan. Contoh dari kasus ini diperlihatkan pada gambar 9.



Gambar 9. Contoh *source code* yang gagal diparsing

**KESIMPULAN**

Telah berhasil dikembagkan sebah bahasa pemrograman (interpreter) yang menggunakan tata bahasa Indonesia sehingga dapat membantu pemrograman pemula Indonesia dalam belajar pemrograman.
Bahasa ini baru mampu menangani struktur dan sederhana maka untuk penelitian lebih lanjut dapat dikembagkan agar mampu menangani struktur dan lanjut msal list-linier.

**Daftar Pustaka**

Aho, Alford V, Ravi Sethi and Jeffry D. Ullman. *Compilers, Principle, Techniques, and Tools*, Addison-Wesley, Reading, Massachusetts.

Bambang Hariyanto, Ir., MT. 2004. *Teori Bahasa Otomata, dan Komputasi serta Terapannya*. Pennerbit Informatika, Bandung

*Grammar Definition Language 1.0 Reference,* Norken Technology, http://www.programmar.com

*ProGrammar Parser Development Toolkit*, Norken Tecnlogy, http://www.programmar.com

Utdirartatmo Firrat. 2005. *Teknik Kompilasi,* Graha Ilmu, Yogyakarta

Wiryana, *Compailer Construction*,http:/wiryana.pandu.org/?id=7

# Bahasa Pemrograman Kinuy

ORIGINALITY REPORT

| 17% | 15% | 3% | 2% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | dominggusngain.blogspot.com<br>Internet Source | 2% |
| 2 | ahrif91.blogspot.com<br>Internet Source | 2% |
| 3 | anzdoc.com<br>Internet Source | 2% |
| 4 | media.neliti.com<br>Internet Source | 2% |
| 5 | iishopeinadream.blogspot.com<br>Internet Source | 1% |
| 6 | alfe123.blogspot.com<br>Internet Source | 1% |
| 7 | "Theory and Applications of Models of Computation", Springer Science and Business Media LLC, 2006<br>Publication | 1% |
| 8 | www.journal.uad.ac.id<br>Internet Source | 1% |

| | | |
|---|---|---|
| 9 | isgb.otago.ac.nz<br>Internet Source | 1% |
| 10 | asnhaba.blogspot.com<br>Internet Source | 1% |
| 11 | archai-rahman.blogspot.com<br>Internet Source | 1% |
| 12 | repository.amikom.ac.id<br>Internet Source | 1% |
| 13 | www.inigema.com<br>Internet Source | 1% |
| 14 | jurnal.umk.ac.id<br>Internet Source | 1% |
| 15 | darwintz.wordpress.com<br>Internet Source | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | Off | Exclude matches | Off |
| Exclude bibliography | Off | | |