TEKNOLOGI INFORMASI

Techno COM JURNAL

ISSN : 1412 - 2693

Jurnal Techno.Com | Vol. 6 | No. 1 | Hal : 1 - 52 | Semarang Februari 2007

Techno.COM JURNAL
Vol. 6 No. 1 Februari 2007

REDAKTUR *menerima sumbangan artikel atau naskah yang belum pernah diterbitkan oleh media lain. Naskah tersebut diketik menurut aturan-aturan penulisan naskah yang terdapat pada cover belakang dalam. Naskah tersebut untuk dimuat. Redaksi berhak menyunting sebagian atau seluruh naskah tetapi tidak merubah maksud dan tujuan naskah tersebut. Naskah yang masuk menjadi hak sepenuhnya bagi redaksi. Naskah yang tidak dimuat tidak dikembalikan kecuali atas permintaan penulis atau disertai perangko balasan.*
Terbit pertama kali pada bulan Februari 2002

Alamat Redaksi

Fakultas Ilmu Komputer
UNIVERSITAS "DIAN NUSWANTORO" SEMARANG
Jl. Nakula I No. 5 - 11 Semarang
Telpon : + 62-24 3517261, 3520165
Faximile : + 62-24 3569684
E-mail : Techno@dinus.ac.id

TEKNOLOGI INFORMASI

Techno Com JURNAL

ISSN : 1412 - 2693

DEWAN REDAKSI :

Jurnal Techno.Com | Vol. 6 | No. 1 | Hal : 1 - 52 | Semarang Februari 2007

Teknologi Informasi

ISSN : 1412 - 2693

Techno.COM JURNAL

Vol. 6 N0. 1 Februari 2007

DAFTAR ISI

DAFTAR ISI

PERANCANGAN DAN IMPLEMENTASI BAHASA PEMPROGRAMAN KINUY (BAHASA PEMPROGRAMAN PRIBUMI)

Tedy Setiadi[1] , Arie Musbandi[2]

ABSTRACT : *This research has developed Kinuy language, which is a programming language that close to an indonesian language and grammatical. 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 used as a semantic analysis. This Interpreter was developed using a Syntax Direct Definition method recursively. The Kinuy Language expectation is for helping a beginer Indonesian Programmer.*

Keyword *: Grammar Definition Language, Recognizer, Interpreter*

PENDAHULUAN

Manusia dapat melakukan interaksi secara efektif dengan menggunakan media bahasa. Bahasa memungkinkan penyampaian gagasan dan pemikiran, tanpa kedua hal itu komunikasi akan sulit terjadi. Dalam lingkungan pemrograman komputer, bahasa pemrograman bertindak sebagai sarana komunikasi antara manusia dan permasalahannya dengan komputer yang dipakai untuk membantu memperoleh pemecahan. Bahasa pemrograman menjembatani antara pemikiran manusia yang sering tidak terstruktur dengan kepastian yang diperlukan oleh komputer untuk melakukan 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 tinggkat tinggi yang telah diciptakan manusia seperti Pascal, BASIC, C ,C++, Java dan lain sebagainya. Mungkin untuk sebagian rekayasawan Indonesia yang sudah familiar dengan bahasa tersebut tidak akan sulit berkomunikasi dengan komputer. Tetapi untuk sebagian rekayasawan Indonesia lain yang baru belajar berkomunikasi dengan komputer akan merasa sulit untuk memahami bahasa tersebut.
Ini dikarenakan tidak satu pun bahasa-bahasa ini mirip dengan Bahasa Indonesia, sebagai contoh instruksi *"for"* yang digunakan untuk *loop*-ing atau perulangan akan sulit dimengerti oleh para pemula pemprogram komputer Indonesia.

Memang ada sekelompok pemrogram Indonesia yang membuat bahasa pemrograman yang mirip dengan Bahasa Indonesia, seperti KILANG (BASIC Indonesia) yang dibuat oleh Prof. Daly S. Naga dan beberapa pemrogram yang membuat bahasa pemrograman "BATAK". Tetapi bahasa-bahasa tersebut sudah hilang jejaknya seiring dengan perkembangan bahasa pemrograman yang sudah maju. Bahasa pemrograman JAVA, BALI, MADURA hanyalah nama saja, tetapi tidak satupun 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 : Alumni Teknik informatika Universitas Ahmad Dahlan Yogyakarta*

PEMBAHASAN

a. Pembuatan Spesifikasi Bahasa

Sebelum menentukan tata bahasa dari bahasa yang akan dibuat hal pertama yang harus dilakukan adalah menentukan spesifikasi bahasa tersebut. Ini dibutuhkan sebagai patokan dalam pembuatan bahasa. Adapun spesifikasi bahasa yang akan dibuat adalah sebagai berikut.

1. *Grammar* dengan pendekatan Bahasa Indonesia
2. Pendeklarasian variabel, konstanta, fungsi dan prosedur.
3. Menangani tipe data *Integer*, *Float*, *String* dan *Boolean*.
4. Sudah *terdapat* fungsi-fungsi bawaan seperti 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 parsing akan menjadi lebih mudah. Dalam pembuatan diagram keadaan dan *grammar* menggunakan perangkat Bantu JFLAPBeta versi 2.0. keunggulan perangkat ini adalah bisa mengkonversi diagram keadaan menjadi *grammar.*

Berdasarkan spesifikasi bahasa, bahasa Kinuy sudah terdapat menangani pembuatan fungsi prosedur, variable, dan konstanta. Artinya bagan program terdiri dari blok-blok fungsi dan fungsi utama. Dan yang harus dipertimbangkan adalah *life scope* dari sebuah variable. Pada bagian ini juga dilakukan penentuan bentuk sintak dari sebuah *statement*. Sebagai contoh bentuk sintak dari *statement* "if" ditentukan sebagai berikut.

```
JIKA <Expresi> MAKA
  [STATEMENTS]
MELAINKAN_JIKA <Expresi> MAKA
  [STATEMENTS]
MELAINKAN
  [STATEMENTS]
AKHIR JIKA
```

Diagram keadaaan atau *state transition diagram* yang termasuk kedalam *Finite automata* adalah model matematika dengan masukkan dan keluaran diskrit. Sistem dapat berada di salah satu dari sejumlah berhingga konfigurasi internal yang disebut *state*. *State* sistem merupakan serangkayan informasi yang berkaitan dengan masukan-masukan sebelumnya yang menentukan prilaku sistem pada masukkan berikutnya. Pembuatan diagram keadaan adalah cara termudah untuk membuat *recognizer* dari bahasa yang kita buat. *Recognizer* membaca string yang kita masukkan dan memberikan keluaran YA jika string tersebut termasuk dalam bahasa yang buat atau TIDAK jika string tersebut tidak termasuk kedalam bahasa

Techno.Com, Vol. 6 No. 1, Februari 2007

Misalkan diagram transisi untuk deklarasi variabel adalah sebagai berikut.

Gambar 1. Diagram tansisi 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 bahasa (*grammar*) bisa didefiniskan secara formal sebagai kumpulan dari himpunan-himpunan variabel, simbol-simbol terminal, simbol awal, yang dibatasi oleh aturan-aturan produksi 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.

Tabel 1. Hirarki Chomsky

Bahasa	Mesin Otomata	Batasa Aturan Produksi
Regular/Tipe 3	*Finite State Automata* (FSA) meliputi *Deterministic Finite Automata* (DFA) dan *Nondeterministic Finite Automata* (NFA)	α adalah sebuah simbol variabel β maksimal memiliki satu simbol variabel yang bila ada terletak di posisi paling kanan
Bebas Kontek/ *Context Free* / Tipe 2	*Push Down Automata* (PDA)	α berupa sebuah simbol variabel
Context Sensitive / Tipe 1	*Linier Bounded Automata*	| α | ? | β |
Unrestricted / *Phase Structure* / *Natural Language* / 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, dan melalui aturan produksi tertentu didefinisikan suatu bahasa yang berhubungan dengan tata bahasa tersebut.

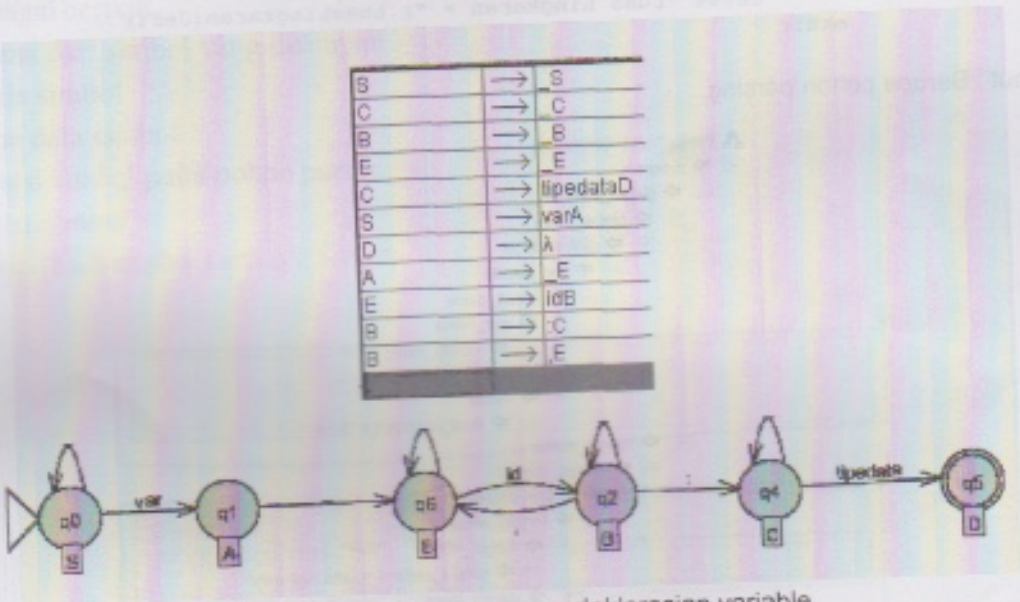
Aturan prosuksi sebuah bahasa dinyatakan dalam bentuk :

$$a \rightarrow b$$

(a menghasilkan b)

Dimana a menyatakan simbol-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 masih bisa diturunkan, sedangkan simbol terminl tidak bisa diturunkan lagi. Simbol terminal biasanya dinyatakan dengan huruf kecil misalnya 'a','b','c' sedangkan simbol non-terminal dinyatakan dengan huruf besar 'A','B',C'

Hasil konversi diagram transisi pendeklarasan variabel ditunjukkan pada gambar dibawah ini.



Gambar 3. Bentuk *grammar* pendeklarasian variable.

c. Implementasi

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

Bentuk GDL dari pendeklarasian variabel adalah sebagai berikut.

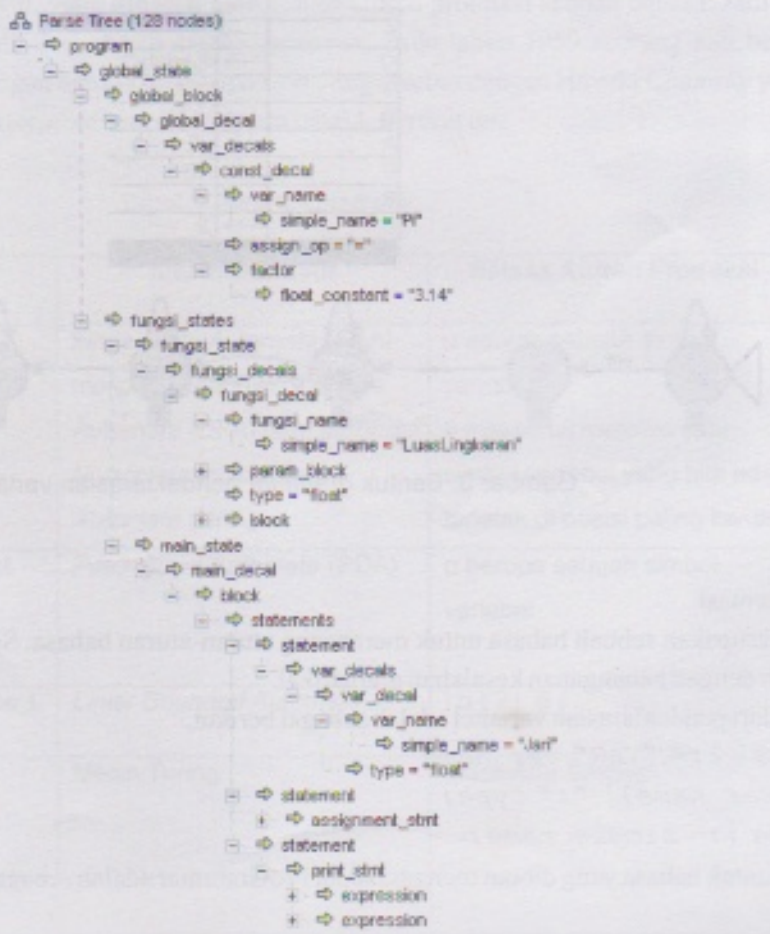
```
var_decal ::= "VAR" var_name
{"," var_name}] ":" type;
var_name ::= simple_name;
```

Hasil parsing untuk bahasa yang dibuat menggunakan ProGrammar 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.



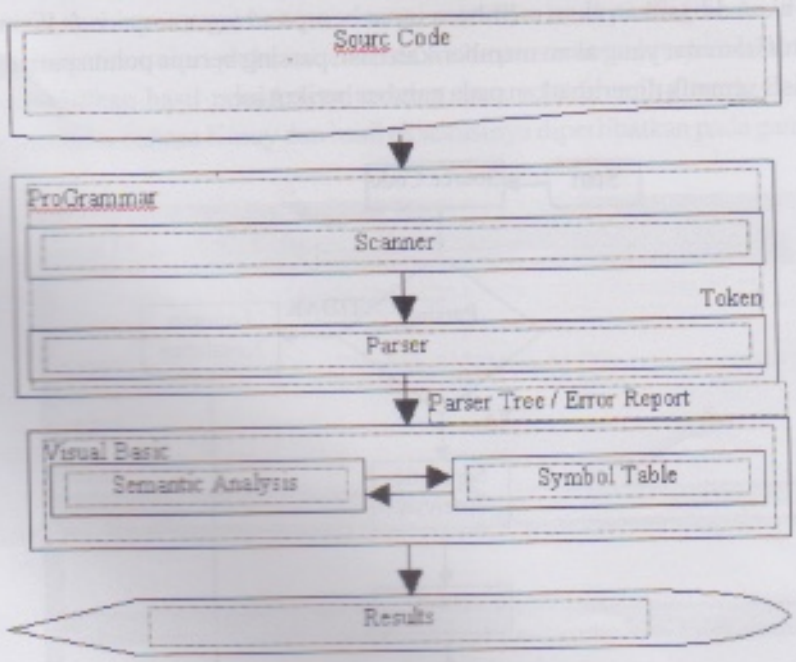
Gambar 4. Pohon Parsing.

Interpreter yang dirancang hanya menentukan makna semantik 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 yang 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 sebuah struktur data yang berfungsi menyimpan semua informasi selama proses interpretasi berlangsung. Semakin kompleks suatu bahasa maka penggunaan struktur data pada tabel simbol juga semakin komplek. Untuk bahasa KINUY menggunakan tabel simbol sederhana dengan memanfaatkan fasilitas pada Visual Basic yang 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 serta *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* hingga ke *node* terbawah atau node tersebut merupakan simbol terminal. sesuai dengan GDL yang telah dibuat. Pembuatan fungsi dan prosedur didasarkan pada GDL yang dihasilkan, nonterminal ditelusuri hingga produksi menyatakan simbol terminal. Jadi GDL bisa dijadikan pengganti dari penganalisis sintak.

Metode *syntax direct definition* parser menerima input yang berupa token dan menghasilkan pohon parser yang kemudian ditentukan makna semantiknya dengan metode ini [1].

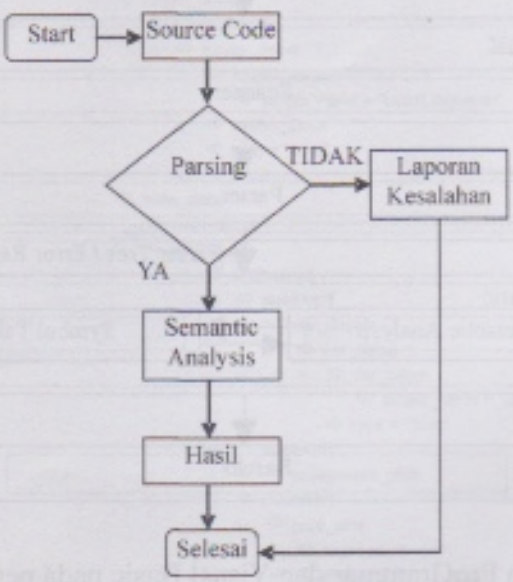
Contoh penerapan *syntax direct definition* dapat dilihat pada tabel 2 dibawah ini.

Tabel 2 *Syntax Direct Diefintion* pada kalkulator sederhana

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

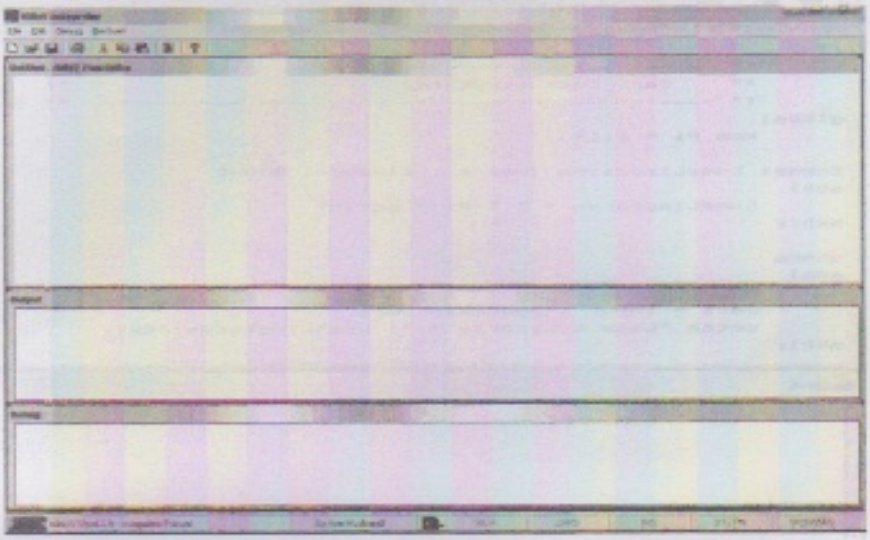
b. Pengembangan Aplikasi

Interpreter yang akan dihasilkan akan membaca input berupa *string source code* kemudian di parsing dengan menggunakan ProGrammar yang akan memberikan hasil parsing berupa pohon parsing. Adapun skema global proses penganalisis sematik diperlihatkan pada gambar berikut ini.



Gambar 6. Global schame pembuatan interpreter.

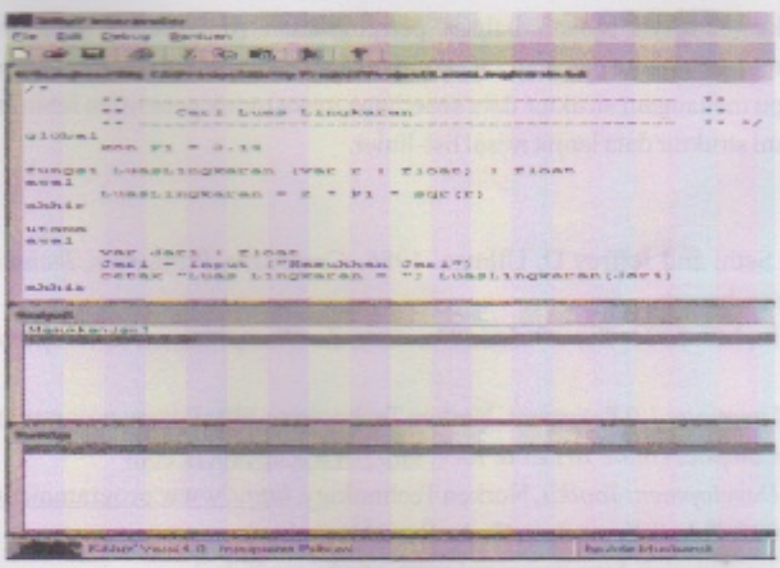
Source code yang di inputkan akan dianalisis oleh parser yang dilakukan ProGrammar apakah sesuai dengan *grammar* atau tata bahasa yang dibuat. Jika *source code* tersebut gagal diparsing yang artinya tidak sesuai dengan tata bahasa yang dibuat maka akan ditampilkan pesan kesalahan, apabila *source code*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



Gambar 7. Form Utama

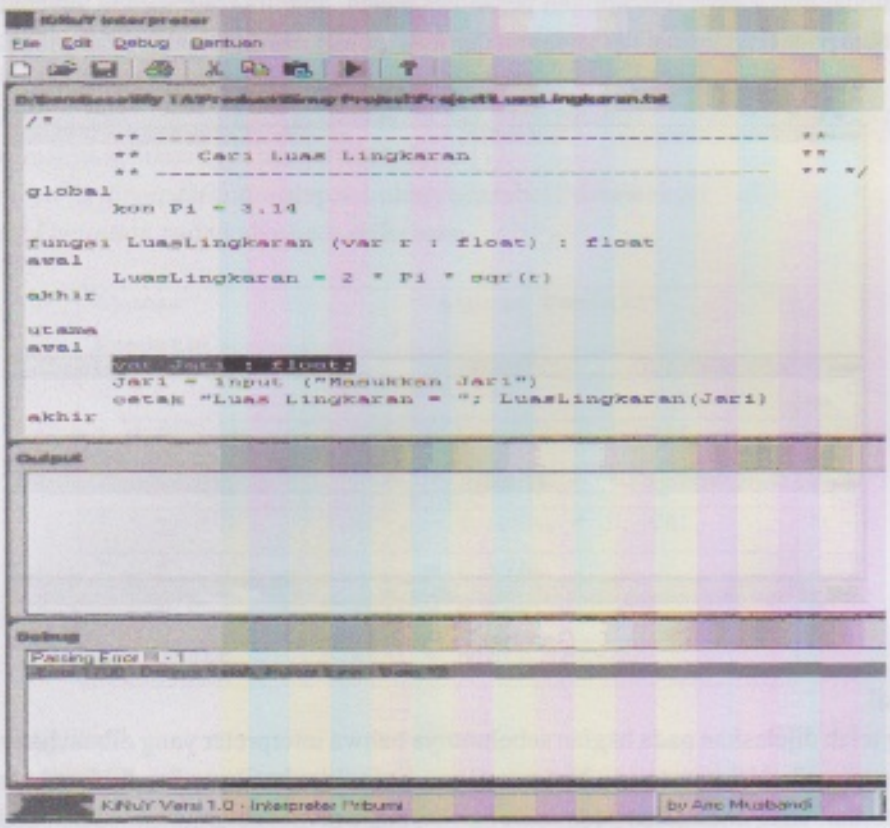
c. Analisis Hasil

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 semantik 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 semantik tidak dilanjutkan dan laporan kesalahan diberikan. Contoh dari kasus ini diperlihatkan pada gambar 9.



Gambar 9. Contoh *source code* yang gagal di parsing.

KESIMPULAN

Telah berhasil dikembangkan sebuah bahasa pemrograman (interpreter) yang menggunakan tata bahasa indonesia sehingga dapat membantu pemrogram pemula di Indonesia dalam belajar pemrograman.

Bahasa ini baru mampu menangani struktur data sederhana maka untuk penelitian lebih lanjut dapat dikembangkan agar mampu menangani struktur data lanjut misal list-linier.

Daftar Pustaka

Aho, Alfred V., Ravi Sethi and Jeffrey D. Ullman. 1986. *Compilers,* Prinicples, *Techniques and Tools*. Addison-Wesley, Reading, Massachusetts

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

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

Niemann, Thomas. *A Compact Guide To Lex & Yacc*, http://www.epeapers.com

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

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

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