LAPORAN AKHIR
PENELITIAN DOSEN PEMULA

TRANSLATOR NOTASI ALGORITMIK UNTUK PENGAJARAN PEMROGRAMAN DASAR

Tahun ke 1 dari rencana 1 tahun

WIJANARTO, S.Sos., M.Kom. 0628027003
AJIB SUSANTO, S.Kom., M.Kom 0615127404

UNIVERSITAS DIAN NUSWANTORO
Desember 2013

HALAMAN PENGESAHAN

Judul Kegiatan	:	TRANSLATOR NOTASI ALGORITMIK UNTUK PENGAJARAN PEMROGRAMAN DASAR
Peneliti / Pelaksana		
Nama Lengkap	:	WIJANARTO
NIDN	:	0628027003
Jabatan Fungsional	:	
Program Studi	:	Teknik Informatika
Nomor HP	:	
Surel (e-mail)	:	wijanarto.udinus@gmail.com
Anggota Peneliti (1)		
Nama Lengkap	:	AJIB SUSANTO M.Kom.
NIDN	:	0615127404
Perguruan Tinggi	:	UNIVERSITAS DIAN NUSWANTORO
Institusi Mitra (jika ada)		
Nama Institusi Mitra	:	
Alamat	:	
Penanggung Jawab	:	
Tahun Pelaksanaan	:	Tahun ke 1 dari rencana 1 tahun
Biaya Tahun Berjalan	:	Rp. 14.500.000,00
Biaya Keseluruhan	:	Rp. 14.852.000,00

Mengetahui
Dekan

(Dr. Abdul Syukur)
NIP/NIK 0686.11.1992.017

Semarang, 4 - 12 - 2013,
Ketua Peneliti,

(WIJANARTO)
NIP/NIK0686.11.2009.354

Menyetujui,
Ketua Lembaga Penelitian

(Y. Tyas Catur P. S.Si., M.Kom)
NIP/NIK 0686.11.1994.046

RINGKASAN

Pemrograman dasar merupakan pondasi utama sesorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Dalam disiplin ilmu komputer menyelesaikan masalah merupakan basis dari perkembangan keilmuan. Algoritma merupakan salah satu teknik untuk memecahkan masalah di bidang pemrograman yang di ekspresikan dalam bahasa pemrograman. Kesulitan utama seseorang dalam mengekspresikan solusi dalam bentuk bahasa formal merupakan masalah tersendiri yang harus di pecahkan, selain pemilihan alat atau aplikasi yang tepat untuk membantunya, bahkan untuk orang dengan latar belakang ilmu komputer.

Dengan demikian yang tujuan jangka panjang yang ingin di capai dalam peneitian ini adalah membuat suatu *Domain Specific Language* (DSL) untuk pengajaran pemrograman dasar. Untuk saat ini, penelitian ini menyajikan suatu alat atau aplikasi untuk mempermudah penyelesaian masalah berbasis notasi algoritmik dalam bidang pengajaran pemrograman dasar bagi mahasiswa di tahun pertama, dalam bentuk suatu editor teks yang dapat mentranslasikan notasi algoritmik ke bahasa c. Alat ini akan membantu seseorang atau mahasiswa untuk dapat mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang di pilihnya. Model notasi algoritmik yang di pilih merupakan model yang sudah pernah diterapkan dan diajarkan di perguruan tinggi. Penelitian ini akan menggunakan metode *Rapid Application Development* (*RAD*) dan *Model View Cotroller* (*MVC*) dalam rangka membangun aplikasi translator notasi algoritmik dalam bentuk editor teks. Untuk mengetahui apakah aplikasi yang di hasilkan mempunyai manfaat nyata, metode eksperimen murni (*pure experimental*) tanpa pretest akan di terapkan untuk mengevaluasi subyek penelitian yang di bagi menjadi dua kelompok, yaitu subyek eksperimen dan subyek kontrol. Sedangkan analisa data di lakukan dengan Uji T, untuk mendapatkan signifikansi perbedaan subyek penelitian terhadap penggunaan translator notasi algoritmik yang di ukur berdasarkan kemampuan (nilai) dan efektifitas (waktu) penyelesaian masalah yang di berikan secara random terhadap subyek penelitian. Hasil penelitian menunjukan terdapat perbedaan kemampuan dan kecepatan menyelesaikan masalah pemrograman pada kelompok eksperimen dan kontrol dengan uji t sebesar 4,638 dengan df =74 diperoleh p sebesar 0,000, dan selisih rata-rata kemampuan sebesar 12.21, sedangkan perbedaan kecepatan di hasilkan nilai t hitung = -4,718 dengan df 74 dengan p sebesar 0,000 dan selisih waktu antara kelompok eksperimen dan kontrol sebesar 11.55.

PRAKATA

Dengan selesainya pembuatan laporan penelitian dosen pemula ini, penulis bersyukur kehadirat Alloh SWT. Laporan penelitian ini merupakan kegiatan lanjutan dari hasil penelitan yang telah di lakukan penulis dalam rangka mencari jawaban atas kegelisahan intelektual di bidang pengajaran pemrograman, khususnya pemrograman dasar.

Pemlihan model pembelajaran pemrograman merupakan inti dari penelitian ini, yang dapat di aplikasikan dalam suatu notasi sederhana untuk pengajaran pemrograman dasar dan pembuatan media atau alat untuk memformulasikan model abstrak menjadi nyata dalam bentuk editor dan translator notasi algoritma ke dalam bahasa formal, dalam hal ini bahasa C.

Tentu saja, capaian ini bukan merupakan akhir dari kegiatan, pengujian dan penyempurnaan model akan secara berkelanjutan akan terus dilakukan oleh penulis guna mencapai suatu titik ideal. Dengan menyeberkan informasi penelitian melalui jurnal atau prosiding, penulis berharap akan mendapatkan masukan yang lebih baik kedepannya, selain itu baik model maupun alat hasil berupa editor akan di lakukan penyemurnaan hingga mendapatkan bentuk yang mudah di pakai dan di pahami oleh pemakai awal.

Akhirnya, penulis tidak lupa juga mengucapkan terima kasih yang sebesar-besarnya kepada para pihak yang telah membantu terselesaikannya laporan ini, terutama, mahasiswa dan asisten peneliti yang terlibat dalam proyek ini. Tak lupa rekan dosen dan segenap jajaran struktural di Fakultas Ilmu Komputer Universitas Dian Nuswantoro yang sangat mendukung kegiatan ini, penulis ingin menyampaikan apresiasi setinggi-tingginya.

Semarang, Desember 2013

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

1.1.Latar Belakang Masalah

Pemrograman dasar merupakan pondasi utama sesorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Sesederhana apapun, masalah yang harus di pecahkan harus dilakukan secara terstruktur dan ilmiah. Dalam dunia ilmu komputer atau teknik informatika langkah-langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma (Blass et.al 2003, Harel et.al 2004). Seperti diketahui algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu, seperti penentuan DNA (Ming, 2005), Teori graph dalam menentukan lintasan terpendek (Kruskal, 1956) dan masih banyak lagi.

Dalam studi yang pernah dilakukan di Afrika Selatan (Cilliers et.al., 2005), keberhasilan pembelajaran pemrograman dasar di pengaruhi oleh, (1) lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman, (2) penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan (3) lingkungan kerja aplikasi harus melindungi siswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Artinya kesulitan utama mempelajari pemrograman di karenakan kesulitan bagaimana menggunakan memahami semantik dari suatu bahasa pemrograman, seperti di jelaskan dalam (Cilliers et.al., 2005).

Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti (Hidayanti, 2007), lebih menyoroti metode pembelajaran dari aspek pedagogik, di mana capaian mahasiswa dalam belajar pemrograman dasar sangat rendah di karenakan rendahnya partisipasi, keaktifan dalam berdiskusi dan bertanya serta menjawab pertanyaan dalam kuliah. Sedangkan peneliti lain (Yuwono, 2009), dalam matakuliah sejenis yaitu komputer

dasar, menyimpulkan (masih dari aspek pedagogik) bahwa metode belajar berbasis pada masalah dapat meningkatkan pemahaman materi dan prestasi mahasiswa, namun hanya efektif di lakukan dalam satu siklus saja.

Dengan demikian menurut hemat kami, dalam rangka mempermudah proses pembelajaran siswa dalam pemrograman dasar diperlukan model yang dapat menyederhanakan struktur dan semantik instruksi, sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman. Model sederhana yang dipakai merupakan suatu translator notasi algoritmik yang secara otomatis dapat menghasilkan suatu bahasa pemrograman tingkat tinggi yang umum (Wijanarto, 2012). Sementara notasi algoritmik yang standar yang diberikan merupakan notasi yang sudah di ajarkan di perguruan tinggi (Liem, 2007).

1.2. Rumusan Masalah

Berdasarkan paparan latar belakang sebelumnya, maka masalah yang di hadapi dan akan di angkat dalam penelitian terdiri dari :

1. Bagaimana menentukan dan membuat model grammar untuk menghasilkan notasi algoritmik ?
2. Bagaimana implementasi model grammar dalam bentuk aplikasi translator notasi algoritmik ke dalam suatu bahasa formal ?
3. Bagaimana menerapkan aplikasi translator notasi algoritmik untuk menyelesaikan masalah di bidang pemrograman dasar?.

1.3. Batasan Masalah

Tentu saja terdapat beberapa batasan dalam rangka menyelesaiakan masalah yang di kemukakan di atas, diantaranya adalah :

1. Model notasi algoritmik yang di pakai berupa notasi yang sudah di ajarkan dalam perkuliahan di perguruan tinggi (Sekolah Teknik Elektro dan Informatika ITB dan Universitas Dian Nuswantoro.
2. Aplikasi hanya mengenerate bahasa pemrograman (formal) prosedural yaitu bahasa C standar.

3. Penanganan terhadap fungsi input dan output sebatas yang umum di pelajari dalam pemrograman dasar.
4. Aplikasi hanya dapat di jalankan dengan Java SE 1.7 ke atas terinstall di komputer.

BAB 2. TINJAUAN PUSTAKA

2.1. Notasi Algoritmik

Dalam penelitian ini, notasi algoritmik yang dimaksud adalah suatu bahasa alami (*psuedocode*) yang mudah di pahami manusia dalam mengeskpresikan suatu solusi atau disain dalam suatu masalah. Walaupun demikian notasi yang dipakai kecenderungannya lebih mendekati ke suatu bahasa formal tertentu untuk mempermudah translasi. Penelitian ini menggunakan notasi algoritma dalam (Liem, 2007) dengan sedikit modifikasi oleh penulis, secara ringkas notasi algoritmik yang di maksud dapat di lihat pada tabel 1 di bawah ini. Selain alasan kemudahan notasi ini sudah di ajarkan pada STEI ITB dan Fakultas Ilmu Komputer, Universitas Dian Nuswantoro.

Tabel 1. Notasi Algoritmik Standar

Notasi Modifikasi	Notasi Standar
Program Berisi setidaknya satu deklarasi: pustaka, makro, type, variabel, konstanta, fungsi dan program utama	**Program <nama-program>** { Spesifikasi teks algoritmik secara umum } **KAMUS** { Definisi konstanta, type, deklarasi variabel, spesifikasi prosedur, fungsi } **ALGORITMA** { Teks algoritma - tidak berada di antara tanda kurung kurawal }
Pustaka **Uses** FILE	Sama
Makro **Def** [ID][As replacement] **IfNotDef** ID **EndDef** **ElseDef**	Tidak Ada
Type (struct, enum, union, array) **Type** [nama = type] [nama : <fields>] [type : <fields>]	**{ Definisi Type Bentukan}** **type** namatype : < elemen1 : type1, elemen2 : type2, ... > **{ Deklarasi Variable }** nmvar1 : namatype nmvar2 : type1 {misal} **{ Akses Elemen }** nmvar2 ← nmvar1.elemen1 nmvar1.elemen2 ← <ekspresi>

	{ Definisi type enumerasi } **type** hari : (senin, selasa, rabu, kamis, jumat, sabtu) **{ Deklarasi variable }** H : Hari **{ Assignment }** H ← senin **{ Definisi type Array}** **type** hari : (senin, selasa, rabu, kamis, jumat, sabtu) **{ Deklarasi variable }** H : Hari **ALGORITMA { Assignment }** H ← senin
Variabel Var Nama:type[<--harga]	**Deklarasi Variable** <nama> : <type> **Inisialisasi/Assignment** <nama> ←<harga> **{ Deklarasi Variabel Array}** nm_array : array [0..nmax-1] of type-array **{ Cara Mengacu Elemen Array }** nm_array_{indeks} **Contoh:** TabInt : array [0..99] of integer TabInt ← 1 X ← $TabInt_{10}$
Konstanta Constant Nama:type [<--harga]	**Deklarasi Constant** constant <nama>:<type>=<harga>
Statements [forStat] [ifstat] [switchstat] [exprs;] [block] [assignStat ;] [proc_callStat] [assignFunction] [statIO ;] [;]	**Assignment Statement** <nama1>←<nama2> <nama> ←<konstanta> <nama> ← <ekspresi> **Expression Statement** nama1 ← nama1 <opr> nama2
Block {* Variabel Constant statements Multi Komentar	Tidak ada

Komentar Whitespace *}	
Input/Output **Input**(text,[exprs]) **Output**(text,[exprs])	input(<list-nama>) **Contoh:** input (X) {x integer} input (X, Y) input (F) {F real} input (s) {s string} output(<list-nama>) **Contoh:** output (X) {x integer} output (X, Y) output (“Contoh output”) output (“Namaku: ”, nama) output (F) {F real} output (CC) {c character}
Kondisi(ifstat, switchstat) **if** <kondisi>**then** <statements>[**else** statments] **Depend on** <nama> <* harga : statements else : harga : statements *>	**Satu Kasus:** if kondisi then aksi **Dua Kasus Komplementer:** if kondisi-1 then aksi-1 else { not kondisi-1 } aksi-2 **Analisa > 2 kasus** depend on nama kondisi-1 : aksi-1 kondisi-2 : aksi-2 ... kondisi-n : aksi-n else : aksi-else
Pengulangan (forStat) nama **Traversal** range [step harga] **Do** [statements] **While** <exprs> **Do** [statements] **Repeat** [statments] **Until** <exprs>	**Pengulangan berdasarkan pencacah:** i traversal [Awal..Akhir] Aksi **Pengulangan berdasarkan kondisi berhenti:** repeat Aksi until kondisi-stop **Pengulangan berdasarkan kondisi ulang:** while (kondisi-ulang) do Aksi {not kondisi-ulang} **Pengulangan berdasarkan dua aksi:** iterate Aksi-1 stop kondisi-stop Aksi-2
Subprogram **Procedure**<nama>(In/Out args) **Function**<nama>(args):type	**function** NAMAF (param1 : type1, param2 : type2, …) → type-hasil { Spesifikasi fungsi } **KAMUS LOKAL** { Semua nama yang dipakai dalam algoritma dari fungsi } **ALGORITMA**

	{ Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus, pengulangan } { Pengiriman harga di akhir fungsi, harus sesuai dengan typehasil } **→ hasil** **Pemanggilan fungsi** nama ← **NAMAF ([list parameter aktual])** output (**NAMAF ([list parameter aktual]))** **procedure** NAMAP (input nama1 : type1, input/output nama2 : type2, output nama3 : type3) { Spesifikasi, Initial State, Final State } **KAMUS LOKAL** { Semua nama yang dipakai dalam BADAN PROSEDUR } **ALGORITMA** { BADAN PROSEDUR } { Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan atau prosedur } **Pemanggilan prosedur** NAMAP(paramaktual1,paramaktual2,paramaktual3)

2.2. Translator

Translator merupakan proses translasi input kode sumber menjadi output program yang dapat di eksekusi (Aho et.al, 2007., Aho and Ullman, 1973), melalui analisa lexical dan pemaknaan semantik, yang terdiri dari parser dan lexer yang di hasilkan oleh grammar dengan string template (ST). Baik lexer dan parser di hasilkan oleh ANTLR dan String Template (ST) (Parr et.al, 2011., Parr, 2010), serta Java sebagai target bahasa generator. Dengan demikian yang bertindak sebagai translator dan generator adalah, parser dan lexer yang di generate dari ANTLR dan ST dalam bentuk class dalam bahasa java.

2.3. Grammar

Grammar merupakan aturan kontekstual suatu sintak dengan terdapat semantik didalamnya dari suatu bahasa formal. Sintak yang di gunakan dalam penelitian ini menggunakan BNF (*Backus-Naur Form*) atau Extended BNF, karena kemudahan notasinya (Aho et.all, 2007., Appel, 1998., Watt et.all 2000), yang terdiri

dari *himpunan berhingga simbol terminal, simbol non terminal, simbol awal dan aturan produksi* $N ::= \alpha \mid \beta$, dimana **N** adalah simbol *non terminal* , ::= berarti *terdiri dari* serta α adalah *string terminal* atau *non terminal* yang mungkin kosong serta simbol | yang berarti *alternatif*, himpunan tadi di sebut sebagai *context-free grammar*, singkatnya *grammar*.

Suatu grammar menentukan abstraksi sintak dalam suatu himpunan *Abstract Syntax Tree* (AST), tiap simpul non terminal dari AST mempunyai label aturan produk yang berlaku dan grammar tidak menghasilkan suatu kalimat untuk simbol terminal yang tidak berperan dalam abstraksi sintak. Dalam implementasi penelitian ini grammar yang di terapkan adalah seperti dalam ANTLR (Par et.al, 2011).

2.4. Pemrograman Dasar

Pemrograman merupakan proses belajar memprogram untuk memecahkan masalah dengan metode dan sistematika tertentu lalu mengeksprsikannya dalam suatu bahasa formal, sehingga belajar pemrograman tidak sama dengan belajar bahasa pemrograman (Liem, 2007). Dari sudut pandang aspek inilah konsep pemrograman yang di pakai dalam penelitian ini menjadi sangat penting. Kerangka pikir mahasiswa di arahkan kepada bagaimana menyelesaikan suatu masalah tanpa harus di repotkan dengan menguasai bahasa tertentu. Notasi algoritmik merupakan salah satu solusi yang mencoba membuat standar bebas bahasa namun tetap menyentuh pada metode dan sistematika penyelesaian masalah di bidang ilmu komputer secara lebih sederhana (natural).

2.5. Text Editor

Hampir seluruh bahasa pemrograman dapat di pastikan menggunakan editor berbasis teks (untuk pemodelan, biasanya menggunakan bahasa pemodelan yang cenderung visual). Sulit untuk menemukan referensi standar mengenai editor text, namum secara umum editor teks adalah suatu lingkungan kerja yang di gunakan untuk mengolah data jenis text dan bukan lainnya (numerik, data terstruktur, data multimedia) (Wikipedia, 2013). Terdapat perbedaan pengolah text biasa, seperti aplikasi pengolah kata (MSWord (Microsoft (TM)), LibreOffice, openOffice (open source)), sementara editor text yang di maksud dalam penelitian ini adalah ditujukan

untuk mengolah teks notasi algoritmik yang akan di translasikan menjadi kode sumber bahasa c (seperti, notepad, wordpad), dan hanya dapat menerima input berupa data teks sederhana, tanpa memproses data visual.

BAB 3. TUJUAN DAN MANFAAT PENELITIAN

3.1 Tujuan Penelitian

Tujuan dalam penelitian yang akan di capai adalah menghasilkan aplikasi atau tool (editor teks) yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C, yang di bagi menjadi :

1. Menentukan dan membuat model grammar untuk menghasilkan notasi algoritmik.
2. Mengembangkan aplikasi translator dari model grammar dari notasi algoritmik yang di pilih untuk menghasilkan bahasa c.
3. Mempermudah penyelesaikan masalah di bidang pemrograman dasar melalui translator notasi algoritmik ke bahasa c.

3.2 Manfaat Penelitian

Manfaat yang di harapkan dari penelitian ini adalah :

1. Mengasilkan model grammar untuk notasi algoritmik
2. Menghasilkan aplikasi translator editor teks yang di gunakan untuk mentranslasikan notasi algoritmik ke bahasa c.
3. Membantu mahasiswa atau orang yang tertarik di bidang pemrograman untuk fokus belajar bagaimana menyelesaikan masalah dengan notasi algoritmik standar yang sederhana tanpa memikirkan bahasa pemrograman yang terkesan rumit dan membingungkan.

3.3 Luaran Penelitian

Luaran dari penelitian ini yaitu menghasilkan suatu aplikasi editor teks yang dapat mentranslasikan notasi algoritmik ke bahasa c yang di beri nama Translator Notasi Algoritmik, dengan kemampuan seperti editor pada umumnya, code completion, Syntax Highlight, penanganan kesalahan sintak, undo dan redo, dan kemampuan lainnya.

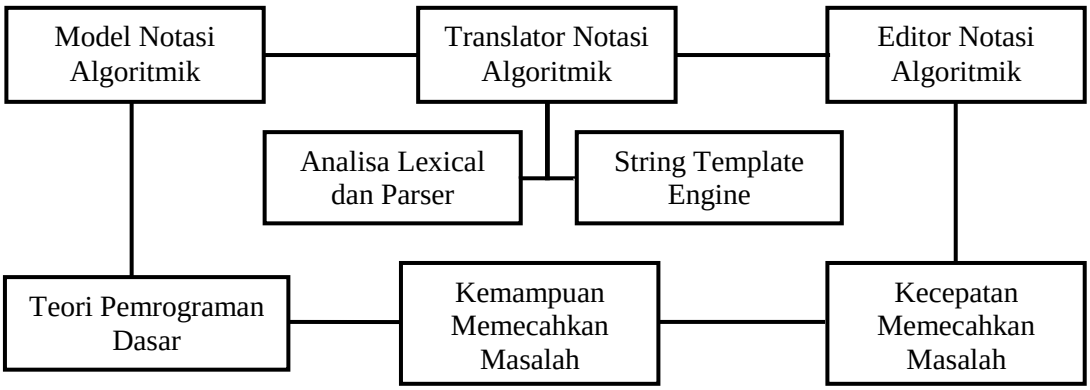
3.4 Kontibusi Penelitian

Kontribusi dari alat ini akan mendukung proses belajar pemrograman dasar dengan model notasi algoritmik yang secara berkelanjutan akan di sempurnakan untuk mencapai hasil optimal. Sehingga juga membawa dampak pada pengembangan

model grammar yang harus di sesuaikan dan dalam ilmu bahasa pemrograman, model ini dapat menjadi studi yang di kenal dengan domain specific language (DSL), yaitu studi bahasa pemrograman (textual atau visual) yang di kembangkan dengan domain yang khusus dan untuk keperluan khusus pula dalam disiplin ilmu komputer.

3.5 Kerangka Pikir

Belajar pemrograman tidak sama dengan belajar program, pemrograman merupakan teknik yang di gunakan untuk menyelesaikan masalah dengan komputer yang di jembatani dengan suatu bahasa natural dalam penyampaiannya. Bahasa natural di sini bukanlah seperti bahasa pada umumnya, misal bahasa Jawa, Sunda atau Inggris, namun merupakan bahasa yang di mengerti manusia dan mudah untuk di terjemahkan menjadi bahasa formal (bahasa formal adalah bahasa yang hanya di mengerti oleh komputer). Dengan demikian dalam belajar pemrograman mahasiswa di harapkan tidak terjebak menggunakan bahasa pemrograman (program) yang cenderung rumit dan sukar di pahami, dan alat untuk membantu mahasiswa dalam belajar pemrograman adalah suatu bahasa natural (Notasi Algoritmik) yang mudah di mengerti dan mudah di terjemahkan oleh komputer dalam rangka menyelesaikan masalah di bidang pemrograman. Translator notasi algoritmik dapat membantu mahasiswa memecahkan masalah di bidang pemrograman dasar tanpa belajar bahasa program yang rumit, seperti di jelaskan pada gambar 1 di bawah ini,



Gambar. 1 Kerangka Pikir

Penggunaan editor notasi algoritmik di harapkan mampu mempermudah mahasiswa dalam memecahkan masalah dengan efektif dalam pembelajaran pemrograman dasar, tanpa belajar bahasa program. Dengan demikian pengujian terhadap editor notasi algoritmik sangat perlu di lakukan. Hasil uji penggunaan editor di harapkan akan memberikan kepastian perbedaana kecepatan pemecahan masalah pemrograman tanpa tahu bahasa program yang di pakai.

3.6 Hipotesa Penelitian

Berdasakan kajian teoritis dan kerangka pikir di atas, maka dapat di ajukan hipotesa sebagai berikut :

1. Hipotesa Nol (Ho)
    a. Tidak ada perbedaan penggunaan translator notasi algoritmik yang signifikan antara kelompok yang menyelesaikan masalah pemrograman dengan menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.
    b. Penggunaan translator notasi algoritmik tidak lebih cepat memecahkan masalah pemrograman dibandingkan dengan tanpa menggunakan translator notasi .
2. Hipotesa Alternatif (Ha)
    a. Terdapat perbedaan kemampuan menyelesaiakan masalah pemrograman yang signifikan antar kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.
    b. Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrograman dibandingkan tanpa menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar.

BAB 4. METODE PENELITIAN

4.1 Teknik Penelitian

Dalam penelitian ini, kami akan menggunakan beberapa teknik umum guna mencapai tujuan penelitian. Penulis akan membagi dua bagian besar yaitu :

1. Metode rekayasa perangkat lunak, yang di gunakan untuk membangun translator notasi algoritmik menjadi suatu aplikasi editor text. Metode rekayasa perangkat lunak yang akan dilakukan terdiri dari penentuan model notasi standar agoritmik, dilanjutkan dengan membuat disain arsitektur sistem, pembangunan sistem dengan metode RAD (Rapid Application Development) dan MVC (Model View Controller).
2. Metode eksperimen, yang di gunakan untuk mengevaluasi apakah hasil implementasi aplikasi bermanfaat untuk mahasiswa yang menggunakannya. Metode evaluasi dalam implementasi yang di pilih adalah metode eksperimen dengan disain kelompok kontrol tanpa pretest (*Posttest Only with Control Group*).

4.2 Model Penelitian

4.2.1 Model Notasi Algoritmik

Menentukan model standar standar notasi algoritmik merupakan jantung dari penelitian ini, di karenakan model ini merupakan kerangka utama dari aplikasi yang akan di hasilkan. Model notasi yang di pilih merupakan model notasi dalam (Wijanarto, 2012). Secara umum arsitektur model grammar yang di pakai adalah seperti dalam gambar 2 sebagai berikut :

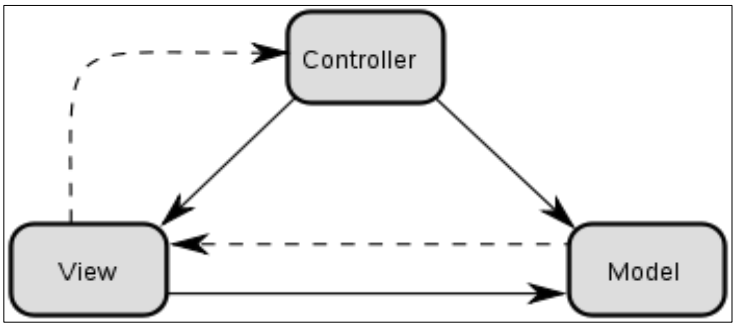
Gambar 2. Model Translator Notasi ke Bahasa C

Seperti dalam (Wijanarto, 2012), model terdiri dari 3 buah langkah yaitu Notasi Algoritmik seperti pada tabel 1, yang berupa bahasa yang mudah di pahami manusia (natural) untuk mengekspresikan disain solusi suatu masalah pemrograman yang

merupakan input yang akan di proses oleh translator dan akan menghasilkan (menggenerate) bahasa formal (bahasa C).

4.2.2 Model View Controller

Metode MVC (Model View Controller) berbasis pada paradigma object oriented. Model atau pendekatan ini pertama kali di sajikan dalam suatu laporan teknis yang di keluarkan oleh Xerox (Reenskaug, 1979) dan dalam perkembangannya pendekatan ini banyak di pakai dalam pengembangan sistem khususnya yang berbasis pada paradigma obyek oriented (Stanchfield, 2009). Metode MVC terdiri dari urutan langkah seperti pada gambar 3 di bawah ini

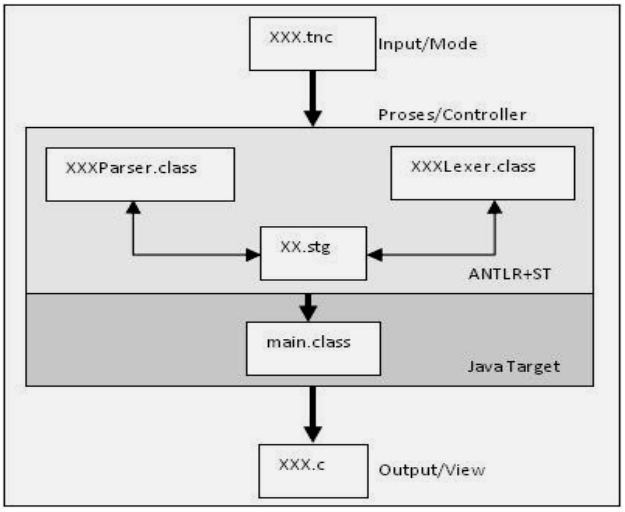


Gambar 3. Model MVC

Model mewakili bagian aplikasi yang menyimpan data dan menyediakan method untuk aksi, ***View*** merupakan bagian aplikasi yang menghasilkan data bagi user dan ***Controller*** adalah bagian yang menerima input dari user dan keperluan modifikasi pada model. Dengan menggunakan metode MVC (Stanchfield, 2009) di samping teknik ini sangat cepat, namun di perlukan keahlian dalam proses pengembangan aplikasi berbasis Java yang multiplatform.. Untuk membuat model grammar, kami memilih model EBNF (Extended Backus Naur Form) yang di generate dengan tool ANTLR. Sedang dalam rangka mentranslasikan grammar ke bahasa kami menggunakan String Template, di samping karena kemudahannya, tool dapat di integrasikan dengan ANTLR. Terakhir untuk menintegrasikan grammar dan template akan di bangun di atas Java yang menghasilkan editor text.

4.2.3 Perancangan dan Pembangunan Arsitektur Sistem

Suatu sistem aplikasi di kembangkan dengan suatu metode atau cara yang beragam, penelitian ini akan menggunakan dua pendekatan dalam mengembangkan aplikasi yaitu Rapid Application Development (RAD) dan Model View Contrller (MVC). Adapun rancangan arsitektur secara umum sebagai kerangka pikirnya adalah seperti gambar 4 sebagai berikut :



Gambar 4. Rancangan Arsitektur MVC Translator Notasi Algoritmik

Input yang berupa file text dalam bentuk notasi standar algoritma akan di baca oleh scanner yang sesuai dengan grammar yang di generate oleh ANTLR. String Template merupakan translator (*hand coded*) notasi ke bahasa yang di spesifikasikan secara simultan saat membuat grammar. Generator notasi, yang menjadi test rig dalam bentuk class akan menghasilkan output bahasa yang valid.

4.2.4 Rapid Application Development

Teknik pembangunan sistem yang di gunakan dengan pendekatan object orinted programming, dengan teknik Rapid application development (RAD). Disamping karena kemudahannya, teknik ini juga sangat cepat dalam membangun sistem skala menengah ke atas. Fase pengembangan sistem dengan metode RAD di bagi menjadi: (1) Fase Planning, untuk menentukan tujuan, fungsionalitas dan scope

yang akan di kerjakan, (2) Fase User Design, yaitu menentukan interface dan bagaimana system akan bekerja dalam bentuk prototype, (3) Fase Construction, Prototype di konversi menjadi aplikasi yang sudah berfungsi, dengan pengkodean dan pengembangan fungsionalitas aplikasi, (4) Fase Cutover, merupakan fase terakhir dimana kegiatan utamnya adalah mencoba pada pemakai dan mendidik para pemakai (Sommerville, 2011).

4.3 Implementasi Aplikasi

Dalam rangka mencapai tujuan penelitian yaitu menghasilkan aplikasi translator notasi agoritmik, kami akan mengimplementasikan penelitian sebagai berikut :

1. Menentukan dan membuat grammar untuk notasi algoritmik untuk menghasilkan suatu kerangka kerja translator notasi algoritmik ke suatu bahasa.
2. Merancang desain *input/output* translator notasi algritmik yang mudah digunakan untuk mahasiswa dengan metode RAD dalam kerangka MVC.
3. Mengimplementasikan rancangan translator notasi algoritmik dalam bentuk editor teks dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno.*

4.4 Pendekatan Penelitian

Pendekatan penelitian yang di pilih dalam penelitian ini adalah penelitian kuantitatif, artinya penelitian yang di lakukan untuk mencari data kuantitatif melalui hasil uji coba eksperimen. Data yang digunakan untuk menganalisis pendekatan kuantitatif ini adalah data berupa angka. Data dalam penelitian kuantitatif adalah berupa angka-angka (Nurgiyantoro, 2001), pendekatan kuantitatif dengan alasan semua gejala yang diamati dapat diukur dan diubah dalam bentuk angka serta dapat dianalisis dengan analisis statistik. Penelitian ini bertujuan untuk menguji suatu teori yang menjelaskan hubungan teori yang ada dengan kenyataan.

Proses pendekatan mengikuti proses berpikir deduktif. Berpikir deduktif, yaitu diawali dengan penentuan konsep yang abstrak berupa teori yang sifat-sifatnya masih umum kemudian dilanjutkan dengan pengumpulan bukti-bukti atau kenyataan untuk pengujian.

4.5 Desain dan Paradigma Penelitian

Dalam metode experimental terdapat tiga jenis rancangan penelitian yaitu (1) Pra-experimen adalah suatu rancangan yang bertujuan mengungkap hubungan sebab-akibat yang melibatkan satu kelompok subyek tanpa kontrol terhadap variabel lainnya(Sugiyono, 2010), (2) eksperimen semu (quasi eksperimental), merupakan rancangan yang mengungkap hubungan sebab akibat dengan melibatkan satu kelompok kontrol dan satu kelompok eksperimen (Sukmadinata et.al., 2007) dan (3) eksperimen murni, rancangan yang melibatkan satu variabel experimen yang berkaitan di beri suatu *treatement* khusus dan satu kelompok kontrol dengan perlakuan yang berbeda setelah menguji hasil (Nasution, 2007).

Penelitan ini menggunakan rancangan eksperimen tanpa pretest (*posttest only with control group*), karena kemudahannya dalam memberi perlakuan khusus terhadap kelompok eksperimen yaitu dengan menguji translator notasi algoritmik untuk mengerjakan masalah yang di berikan dan melakukan kontrol terhadap kelompok lainnya. Penggunaan desain ini hanya melakukan postes baik terhadap kelompok experimen maupun terhadap kelompok kontrol. Penempatan subjek dalam experimen maupun terhadap kelompok kontrol kelompok masing-masing dilakukan dengan penugasan acak, tabel 2 di bawah ini menjelaskan disain kelompok tanpa pretest.

Tabel 2. Disain Subyek Acak Kelompok Eksperimen dan Kontrol

	Kelompok	**Variabel terikat**	**Posttest**
A	Eksperimen	X	Ye
A	Kontrol	-	Yk

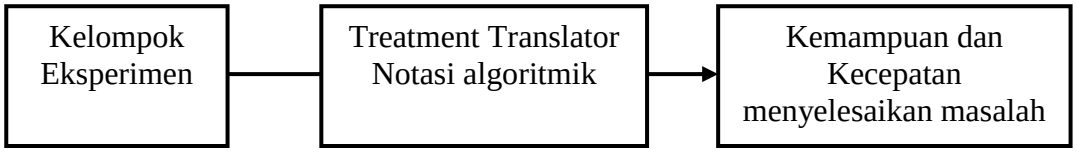
Tabel 2 akan menugaskan setiap subyek pada kelompok eksperimen dan kelompok kontrol secara acak (A). Kemudian hanya melaksanakan perlakuan pada kelompok eksperimen (tanda X) sedangkan pada kelompok kontrol tidak di perlakukan (tanda -), dan kemudian akan melaksanakan posttest pada kelompok eksperimen (Ye) dan kelompok kontrol (Yk), untuk menentukan perbedaan rata-rata nilai yang di peroleh pada Ye dan Yk dengan metode statistik (Uji T), sehingga dapat menentukan signifikansi perbedaan kedua kelompok tersebut.

4.5.1 Paradigma Penelitian

Paradigma penelitian adalah pola pikir yang menunjukkan hubungan antara variabel yang akan diteliti yang sekaligus mencerminkan jenis dan jumlah rumusan masalah yang perlu dijawab melalui penelitian, teori yang digunakan untuk merumuskan hipotesis, jenis dan jumlah hipotesis, dan teknik analisis statistik yang akan digunakan (Sugiyono, 2010).

Paradigma yang digunakan dalam penelitian ini adalah paradigma sederhana. Paradigma sederhana terdiri atas satu variabel independen dan dependen (Sugiyono, 2010). Paradigma dalam penelitian ini dapat digambarkan pada gambar 5 dan 6, sebagai berikut.

a. Paradigma Kelompok Eksperimen



Gambar 5 Paradigma Kelompok Eksperimen

b. Paradigma Kelompok Kontrol

Gambar 6 Paradigma Kelompok Kontrol

Dari gambar paradigma penelitian di atas, variabel penelitian yang telah ditetapkan dikenal paska-uji dengan pengukuran penggunaan tanpa prates. Pembelajaran tanpa menggunakan translator notasi pada kelompok eksperimen dan pembelajaran dengan menggunakan translator notasi algoritmik untuk kelompok kontrol. Setelah itu, kedua kelompok tersebut dikenai pengukuran dengan menggunakan pascates.

4.6 Variabel Penelitian

Dalam penelitian eksperimen kuasi, (Arikunto, 2006) mengatakan bahwa objek penelitian atau apa saja yang menjadi titik perhatian suatu penelitian disebut

sebagai variabel. Variabel dalam penelitian ini diklasifikasikan menjadi dua, yaitu variabel bebas (X) dan variabel terikat (Y). Variabel dalam penelitian ini dapat dilihat sebagai berikut.

1. Variabel bebas

Variabel bebas merupakan variabel yang mempengaruhi atau yang menjadi sebab perubahannya atau timbulnya variabel terikat (Sugiyono, 2010). Variabel bebas dalam penelitian ini adalah penggunaan translator notasi algoritmik untuk menyelesaikan masalah pembelajaran pemrograman dasar.

2. Variabel terikat

Variabel terikat adalah variabel yang dipengaruhi atau yang menjadi akibat, karena adanya variabel bebas. Variabel terikat dalam penelitian ini adalah kemampuan dan kecepatan mahasiswa dalam menyelesaikan masalah pembelajaran pemrograman dasar.

4.7 Tempat dan Waktu Penelitian

1. Tempat Penelitian

Penelitian ini mengambil lokasi di Laboratorium Dasar Fakultas Ilmu Komputer Universitas Dian Nuswantoro Semarang , dengan studi banding di Laboratosium Dasar STEI ITB Bandung. Kelas yang di ambil sebagai obyek penelitian adalah mahasiswa tahun pertama, Program studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang.

2. Waktu Penelitian

Pelaksanaan penelitian dilakukan pada jam praktikum mata kuliah pemrograman dasar supaya siswa mengalami suasana pembelajaran seperti biasanya. Penelitian ini dilakukan pada tanggal 22 Oktober 2013 sampai dengan tanggal 12 November 2013. Penelitian ini dilakukan dalam beberapa tahap, yaitu: 1) tahap pascatest 1 kelompok kontrol dengan menggunakan translator notasi algoritmik, 2) tahap pascatest 2 kelompok eksperimen dengan menggunakan translator algoritmik. Proses pengumpulan data dapat di amati melalui tabel 3 di bawah ini.

Tabel 3. Jadwal Pengambilan Data Penelitian

No	Hari / Tanggal	Kegiatan	Kelas	Sesi ke-
1	Selasa, 22 Oktober 2013	Pascates 2 Kelompok Eksperimen	A11.4101	5
2	Selasa, 22 Oktober 2013	Pascates 1 Kelompok Kontrol	A11.4110	2
3	Selasa, 29 Oktober 2013	Pascates 2 Kelompok Eksperimen	A11.4101	3
4	Selasa, 29 Oktober 2013	Pascates 1 Kelompok Kontrol	A11.4110	6
5	Selasa, 12 November 2013	Pascates 2Kelompok Eksperimen	A11.4101	2
6	Selasa, 12 November 2013	Pascates 1 Kelompok Kontrol	A11.4110	5

4.8 Populasi dan Sampel Penelitian

Dalam Arikunto (Arikunto,2006) menyatakan bahwa populasi adalah kesuluruhan subjek penelitian. Begitu juga dalam Sugiyono (Sugiyono, 2011), populasi adalah wilayah generalisasi yang terdiri atas subyek atau obyek yang mempunyai kualitas dan karakteristik tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya. Populasi dalam penelitian ini adalah mahasiswa tahun pertama kelompok A11.4101 dan A11.4110 program studi Teknik Informatika FIK UDINUS. Terdapat kelas pararel pada mahasiswa pemrograman dasar tahun pertama di program studi teknik infrmatika FIK UDINUS, yaitu sebanyak 15, masing-masing kelas terdiri dari 30 sampai 40 siswa.

Sampel adalah bagian dari jumlah dan karakteristik yang dimiliki oleh populasi. Bila populasi besar dan peneliti tidak mungkin mempelajari semua yang ada pada populasi, misalnya karena keterbatasan dana, tenaga, dan waktu, maka peneliti dapat menggunakan sampel yang diambil dari populasi itu (Sugiyono, 2010). Pada penelitian ini dibutuhkan sampel sebesar dua kelas, sampel yang nantinya akan diambil adalah dua kelas yang harus benar-benar representatif. satu kelas sebagai kelompok eksperimen yaitu pada kelas A11.4101, dan satu kelas sebagai kelompok kontrol, yaitu kelas A11.4110.

4.9 Alat dan Teknik Pengumpulan Data

1. Instrumen Pengumpulan Data

Teknik pengumpulan data dilakukan dengan tes. Tes dilakukan pada saat praktikum dasar pemrograman. Selanjutnya, pascates digunakan untuk mengetahui prestasi kemampuan akhir mahasiswa. Pascates dilakukan untuk mengetahui

kemampuan dan kecepatan mahasiswa setelah mendapat perlakuan. Pascates ini dilakukan pada kelompok kontrol dan kelompok eksperimen. Pembelajaran dilaksanakan di dalam kelas dan materi yang diambil adalah Fungsi dan Prosedur dalam mata pemrograman dasar.

2. Pengembangan Instrumen Penelitian

Instrumen adalah suatu alat yang digunakan untuk mengukur fenomena alam maupun sosial yang diamati. Secara spesifik fenomena tersebut adalah variabel yang diamati. Instrumen yang digunakan dalam penelitian ini berupa tes menyelesaiakan masalah pemrograman dasar yang berfungsi mengukur kemampuan dan kecepatan mahasiswa dalam menyelesaikan masalah emrograman dasar pada mahasiswa tahun pertama di program studi teknik informatika FIK UDINUS. Tes ini berupa menyelesaiakan masalah sederhana yang di kerjakan kelompok kontrol dan kelompok eksperimen. Berikut kisi-kisi instrumen tes Fungsi dan Prosedur.

Tabel 4. Kisi-kisi Instrumen Tes Masalah Fungsi dan Prosedur

No	Pokok Bahasan	Indikator	Jenis
1	Deklarasi Prototype Fungsi dan prosedur	Mahasiswa mampu mengidentifikasi dan mendefinisikan spesifikasi kebutuhan fungsi dan prosedur (Nama, argumen input/output, hasil balik dan tipe fungsi)	Definisi dan Spesifikasi Fungsi atau prosedur
2	Algoritma penyelesaian masalah dalam badan fungsi dan prosedur	Mahasiswa mampu membuat penyelesaian masalah dengan notasi algoritmik untuk menghasilkan penyelesaian yang efisien dan cepat.	Algoritma penyelesaian yang tepat
3	Hasil balik fungsi dan output prosedur	Mahasiswa mampu menghasilkan solusi output yang valid untuk input yang valid pula.	Output penyelesaian yang valid

4	Kecepatan waktu pengerjaan	Mahasiswa dapat menyelesaikan permasalahan pemrograman dasar dalam waktu yang di batasi	Waktu penyelesaian masalah dalam 60 menit

Adapun pedoman penilaian yang dipakai untuk instrumen penelitian ini berupa spesifikasi dan defnisi permintaan masalah yang harus di buat, output yang di inginkan dari masalah yang ada, penggunaan standar dasar algoritma, tipe data dan efisiensi penyelesaian masalah (algoritma) yang berkaitan dengan model matematika umum. Perancangan pedoman penilaian ini juga telah melalui proses *expert judgement. Expert judgement* dalam penelitian ini adalah team laboratorium pemrograman yang di pimpin oleh Wijanarto, M.Kom dan tim laboratrium dasar FIK selaku dosen pemrograman dasar. Pedoman penilaian masalah pemrograman dasar dapat dilihat sebagai berikut.

Tabel 5. Rubrik Penilaian Masalah Pemrograman Dasar

Keterangan Penilaian		**Nilai**	**Skala**
Prototype	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi sangat jelas	40-50	5
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi cukup jelas	30-39	4
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi kurang jelas	20-29	3
	• Menulis Fungsi atau Prosedur dengan spesifikasi dan definisi tidak jelas	10-19	2
	• Tidak menulis Fungsi atau Prosedur	0 - 9	1
Algoritma	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang sangat	40-50	5
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang cukup	30-39	4
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang kurang	20-29	3
	• Menggunakan struktur algoritma, tipe data yang tepat, variabel yang tidak	10-19	2
	• Tidak menggunakan struktur dasar algoritma	0 - 9	1
Validitas Ouput	• Menghasilkan validitas output sangat tepat dan presisi	40-50	5
	• Menghasilkan validitas output cukup tepat dan presisi	30-39	4
	• Menghasilkan validitas output kurang tepat dan presisi	20-29	3
	• Menghasilkan validitas output tidak tepat dan presisi	10-19	2
	• Tidak menghasilkan output	0 - 9	1
Waktu	• Menyelesaikan dalam waktu sangat cepat, kurang dari 30	40-50	5
	• Menyelesaikan dalam waktu cukup cepat, antara 30 hingga 40 menit	30-39	4
	• Menyelesaikan dalam waktu kurang cepat, antara 40 hingga 50 menit	20-29	3
	• Menyelesaikan dalam waktu tidak cepat, antara 50 hingga 60 menit	10-19	2
	• Tidak selesai dalam waktu yang di tentukan, lebih dari 60 menit	0 - 9	1

4.10 Uji Validitas Instrumen

Validitas yaitu suatu ukuran yang menunjukkan tingkat kevalidan suatu instrumen. Suatu instrumen yang valid mempunyai tingkat validitas tinggi dan begitu juga sebaliknya apabila instrumen tidak valid maka validitasnya rendah (Arikunto, 2006).

Dalam penelitian ini, instrumen yang digunakan adalah tes menulis, maka validitas yang digunakan adalah validitas isi (*content validity*). Validitas ini digunakan untuk mengetahui seberapa instrumen tersebut telah mencerminkan isi yang dikehendaki. Soal tes masalah pemrograman sesuai dengan materi yang digunakan dalam Kurikulum Program Studi Teknik Informatika FIK UDINUS dan STEI ITB, khususnya matakuliah dasar pemrograman tahun pertama. Selain itu, instrumen yang digunakan dalam pembelajaran pemrograman dasar di lakukan konsultasikan terlebih dahulu pada ahlinya (*expert judgement*).

4.11 Uji Reliabilitas Instrumen

Reliabilitas menunjuk pada suatu pengertian bahwa suatu instrumen dapat dipercaya untuk digunakan sebagai alat pengumpul data (Arikunto, 2006:154). Kriteria keterpercayaan tes menunjuk pada pengertian tes mampu mengukur secara konsisten sesuatu yang akan diukur dari dari waktu ke waktu.

Koefisien reliabilitas dalam penelitian ini menggunakan penghitungan rumus *Alpha Crombach*. Penghitungan rumus tersebut menggunakan bantuan komputer program SPSS 20. Pengujian reliabilitas dilaksanakan sebelum diberikan masalah pada kelas eksperimen dan kontrol dimulai. *Alpha Cronbach* dapat dipergunakan dengan baik untuk instrumen yang jawabannya berskala, maupun jika dikehendaki yang bersifat dikhotomis (Nurgiyantoro, 2009), Oleh karena itu, *Alpha Cronbach* juga dipergunakan untuk menguji reliabilitas pertanyaan-pertanyaan atau soal-soal esai. Pada penelitian ini ada dua instrumen yang harus diukur reliabilitasnya, yaitu solusi masalah dan waktu pengerjaan solusi. Hasil perhitungan koefisiensi reliabilitas dengan *Alpha Cronbach* tersebut diinterpretasikan dengan tingkat keandalan koefisiensi korelasi sebagai berikut.

Tabel 6. Rubrik Penilaian Masalah Pemrograman Dasar

Antara 0,800 sampai 1000 adalah sangat tinggi
0,600 sampai 0,799 adalah tinggi
0,400 sampai 0,599 adalah cukup
0,200 sampai 0,399 adalah rendah

0,000 sampai 0,179 adalah sangat rendah (Arikunto, 2006). Berdasarkan perhitungan yang telah dilakukan dengan bantuan program SPSS 15.0 di dapatkan koefisien reliabilitas isian masalah pemrograman dasar sebesar 0,883. Berdasarkan hasil tersebut, maka dapat disimpulkan bahwa instrumen tersebut memiliki indeks reliabilitas yang sangat tinggi. Hasil perhitungan dapat dilihat pada lampiran.

4.12 Prosedur Penelitian

Prosedur penelitian yang digunakan dalam penelitian ini adalah sebagai berikut.

4.12.1. Pelaksanaan (*Treatment*)

Kedua kelompok dianggap memiliki kondisi yang sama dan tanpa diberikan prates, dan selanjutnya akan diadakan *treatment* (perlakuan). Tindakan ini dengan menggunakan translator notasi algoritmik, mahasiswa dan peneliti. Peneliti berperan sebagai pengamat yang mengamati secara langsung proses perlakuan. Pada tahap ini, ada perbedaan perlakuan antara kelompok eksperimen dan kelompok kontrol. Dalam pembelajaran penyelesaian masalah pemrograma dasar, kelompok eksperimen diberi perlakuan dengan menggunakan translator notasi algoritmik, sedangkan kelompok kontrol tidak mendapatkan perlakuan tersebut. Adapun tahap-tahap eksperimen adalah sebagai berikut.

a) Kelompok Eksperimen

Kelompok eksperimen akan diberi perlakuan dengan menggunakan translator notasi algoritmik sebanyak tiga perlakuan. Mahasiswa menyelesaiakan masalah dengan menggunakan translator notasi algoritmik. Berikut langkah-langkah pembelajaran penyelesaian masalah dalam pemrograman dasar pada kelompok eksperimen.

1) Perlakuan Pertama

Baik kelompok eksperimen maupun kontrol di asumsikan dalam keadaan yang sama (tanpa pretest) dan mendapat perlakuan yaitu tanpa menggunakan translator notasi algoritmik. Proses *treatment* dengan tidak menggunakan translator notasi algoritmik melalui langkah- langkah sebagai berikut.

(a) Mahasiswa mendapat soal dan memahami spesifikasi dan definisi soal.
(b) Mahasiswa membuat disain kasar penyelesaian soal mlealui kertas.
(c) Mahasiswa mulai mengerjakan dengan kompiler standar
(d) Mahasiswa melakukan debugging dan pengujian.
(e) Peneliti mencatat hasil dan waktu penyelesaian pekerjaan mahasiswa.

2) Perlakuan kedua

Dalam pertemuan kedua kelompok eksperimen mendapatan perlakuan dengan menggunakan translator notasi algoritmik. Proses *treatment* untuk kelompok eksperimen dengan menggunakan translator notasi algoritmik adalah sebagai berikut.

(a) Mahasiswa mendapat soal dan memahami spesifikasi dan definisi soal.
(b) Mahasiswa membuat disain kasar penyelesaian soal mlealui kertas.
(c) Mahasiswa mulai mengerjakan dengan translator notasi algoritmik
(d) Mahasiswa melakukan debugging dan pengujian.
(e) Peneliti mencatat hasil dan waktu penyelesaian pekerjaan mahasiswa.

b. Kelompok Kontrol

Kelompok kontrol mendapatkan pembelajaran masalah pemrograman dasar yang dilaksanakan tanpa menggunakan translator notasi algoritmik (perlakuan 1) tetapi menggunakan apa yang biasanya digunakan sebelumnya yaitu kompiler standar dalam menyelesaikan masalah di bidang pemrograman dasar.

4.12.2. Pengukuran Sesudah Eksperimen *(Post-Experiment Measurement)*

Langkah mahasiswa setelah mendapat perlakuan, kelompok eksperimen dan kelompok kontrol diberi pascates. Tes ini bertujuan untuk melihat pencapaian peningkatan kemampuan memecakan masalah pemrograman dasar setelah diberi perlakuan dengan menggunakan translator notasi algoritmik dan yang tidak diberi perlakuan dengan menggunakan translator notasi algoritmik. Pascates juga

digunakan untuk membandingkan nilai yang dicapai mahasiswa sama, meningkat, atau menurun.

4.13 Teknik Analisis Data

Dalam penelitian ini, analisis data menggunakan rumus Uji-t dan gain skor. Uji-t dimaksudkan untuk menguji rata- rata hitung di antara kelompok-kelompok tertentu (Nurgiyantoro, 2009). Uji-t dalam penelitian ini digunakan untuk menguji perbedaan rata-rata hitung, apakah ada perbedaan signifikan atau tidak antara kelompok eksperimen dengan kelompok kontrol. Syarat data bersifat signifikan apabila nilai p lebih kecil daripada taraf signifikansi 5%.

Gain skor adalah selisih *mean* treatmen dan pascates masing-masing kelompok kontrol dan eksperimen. Gain skor digunakan untuk mengetahui adanya peningkatan atau penurunan skor, untuk mengetahui kecepatan dari penggunaan translator notasi algoritmik dala menyelesaikan masalah pemrograman. Namun, sebelum dilakukan pengujian terhadap hipotesis maka akan dilakukan uji persyaratan analisis terlebih dahulu, yaitu uji normalitas sebaran dan uji homogenitas.

4.13.1. Uji Persyaratan Analisis Data

a. Uji Normalitas Sebaran Data

Uji normalitas bertujuan untuk mengetahui apakah segala yang diselidiki memiliki distribusi normal atau tidak. Uji normalitas ini menggunakan teknik statistik Kolmogorov-Smirnov (Uji K-S). Interpretasi hasil uji normalitas dengan melihat nilai *Asymp. Sig. (2tailed).* Adapun interpretasi dari uji normalitas adalah sebagai berikut.

1. Jika nilai *Asymp. Sig. (2tailed)* lebih besar dati tingkat *Alpha 5%* (*Asymp. Sig. (2tailed)* > 0,05) dapat disimpulkan bahwa data berasal dari populasi yang berdistribusi normal.
2. Jika nilai *Asymp. Sig. (2tailed)* lebih kecil dari tingkat *Alpha 5%* (*Asymp. Sig. (2tailed)* < 0,05) dapat disimpulkan bahwa data berasal dari populasi yang berdistribusi tidak normal.

b. Uji Homogenitas Varian

Uji homogenitas bertujuan untuk mengetahui apakah sampel yang diambil dari populasi memiliki varian yang sama atau tidak menunjukkan perbedaan yang signifikan satu sama lain. Interpretasi hasil uji homogenitas dengan melihat nilai *Sig*. Adapun interpretasinya sebagai berikut.

a. Jika signifikan lebih kecil dari 0,05 (*Sig*. < *alpha*), maka varian berbeda secara signifikan (tidak homogen).
b. Jika signifikan lebih besar dari 0,05 (*Sig*. > *alpha*), maka varian berbeda secara signifikan (homogen).

4.13.2. Analisis Data

Teknik analisis data yang digunakan untuk menguji hipotesis dalam penelitian ini adalah Uji-t (*t-test*). Uji-t untuk menguji apakah nilai rata-rata dari kedua kelompok tersebut memiliki perbedaan yang signifikan teknik analisis data dilakukan dengan menggunakan komputer program SPSS 20. Interpretasi hasil Uji-t dengan melihat nilai *Sig. (2tailed),* kemudian dibandingkan dengan tingkat signifikasi 0,050. Adapun interpretasi dari Uji-t adalah sebagai berikut.

a. Jika nilai *Sig. (2-tailed)* lebih besar dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* > 0,05), maka dapat disimpulkan bahwa tidak terdapat perbedaan yang positif dan signifikan antara mahasiswa yang menggunakan translator notasi algoritmik dibanding dengan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam pemecahan masalah pemrograman dasar.
b. Jika nilai *Sig. (2-tailed)* lebih kecil dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* < 0,05), maka dapat disimpulkan bahwa terdapat perbedaan yang positif dan signifikan antara mahasiswa yang menggunakan translator notasi algoritmik dibanding dengan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam pemecahan masalah pemrograman dasar. Setelah dilakukan Uji-t, dapat diambil kesimpulan bahwa;
    1. Jika nilai *Sig. (2-tailed)* lebih besar dari tingkat signifikasi 0,05 (*Sig. (2- tailed)* > 0,05), maka dapat disimpulkan bahwa penggunaan translator notasi algoritmik tidak lebih cepat

dibandingkan dengan yang tanpa menggunakan translator notasi algoritmik dalam pembelajaran pemrograman dasar.

2. Jika nilai *Sig. (2-tailed)* lebih kecil dari tingkat signifikasi 0,05 (*Sig. (2-tailed)* < 0,05), maka dapat disimpulkan bahwa penggunaan translator notasi algoritmik lebih cepat dibandingkan dengan yang tanpa menggunakan translator notasi algoritmik dalam pembelajaran pemrograman dasar.

4.14 Hipotesis Statistik

Hipotesis statistik disebut juga hipotesis nihil (Ho). Hipotesis ini menyatakan tidak terdapat perbedaan yang signifikan kecepatan menyelesaikan masalah pemrograman dasar kelas eksperimen yang diajar dengan menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ho = μ1 :μ2
Ha = μ1≠μ2

Ho = Tidak terdapat perbedaan yang signifikan kecepatan menyelesaiakan masalah pemrograman dasar antara kelas eksperimen yang diajar dengan menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ha = Terdapat perbedaan yang signifikan kecepatan menyelesaikan masalah perograman dasar antara kelas eksperimen yang diajar menggunakan translator notasi algoritmik dengan kelas kontrol yang diajar tanpa menggunakan translator notasi algoritmik.

Ho = μ1:μ2
Ha = μ1>μ2

Ho = translator notasi algoritmik tidak efektif digunakan sebagai media pemecahan masalah pmrograman dasar pada mahasiswa tahun pertama progrm studi teknik informatika FIK UDINUS.

Ha = translator notasi algoritmik dapat cepat menyelesaikan masalah yang digunakan sebagai media pembelajaran pemrograman dasar pada mahasiswa tahun pertama program studi teknik informatika FIK.

BAB 5. HASIL YANG DICAPAI

5.1 Hasil Penelitian

Dalam penelitian ini, terdapat dua hasil yang akan di paparkan, yang terdiri dari hasil penelitian pengembangan sistem (perekayasaan) editor translator notasi algoritmik dan hasil penelitian eksperimen yang bertujuan untuk mendiskripsikan perbedaan penyelesaian masalah pemrograman dasar dengan translator notasi algoritmik. Perbedaan yang ingin di lihat adalah pada mahasiswa yang diberi perlakuan atau kelompok eksperimen dengan menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar dan mahasiswa yang tidak menggunakan translator notasi algoritmik dalam menyelesaiakan masalah pemrograman dasar yang di sebut kelompok kontrol. Dengan demikin penelitian ini juga dapat menguji kecepataan penyelesaian masalah pemrograman dasar dengan translator notasi algoritmik. Penggunaan data di peroleh dari posttest baik pada kelompok eksperimen dan kontrol.

5.1.1. Hasil Penelitian Perekayasaan

Terdapat tiga tahap sebagai hasil dari pengembangan sistem yang di pilih, yaitu *pertama*, hasil pembuatan grammar untuk notasi algoritmik untuk menghasilkan suatu kerangka kerja translator notasi algoritmik ke suatu bahasa. *Kedua*, implementasi rancangan translator notasi algoritmik dalam bentuk editor teks dengan pendekatan RAD dan MVC, yang di implementasikan dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno.*

5.1.1.a. Grammar dan String Template Translator Notasi Algoritmik

Grammar di bangun dengan menggunakan library ANTLR dengan editor ANTLRWorks, dan di tulis langsung (hand coded). Format grammar (Parr, 2007) yang di pakai dalam peelitian ini adalah sebagai berikut :

```
grammarType grammar name;
«optionsSpec»
«tokensSpec»
«attributeScopes»
«actions»
```

```
/** doc comment */
rule1 : ... | ... | ... ;
rule2 : ... | ... | ... ;
   ...
```

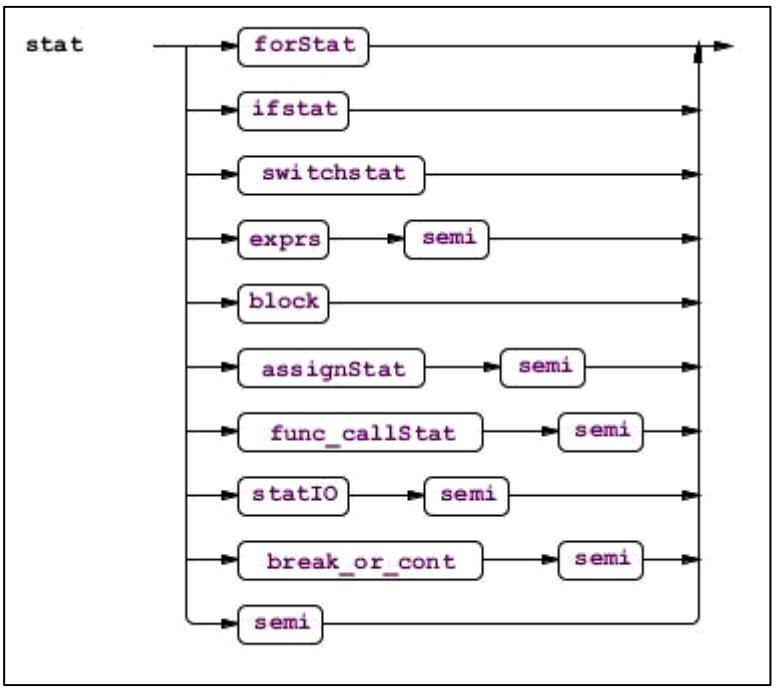
Potongan grammar file yang di buat bernama *algritmik.g* seperti dalam gambar 7 berikut, yang selengkapnya ada di lampiran :

```
grammar Algoritmik;
options {
      backtrack=true;
      memoize=true;
      k=2;
      language=Java;
      output=template;
}

program
    : declaration+
    ;
.....
forStat    :
  ID* 'Traversal' ('[' s=logCond range e=logCond ']')*
      ('Step' exprs_or_assignStat)* 'Do' block ';'
  |   'Repeat' block 'Until' '('(logCond)*')' ';'
  |   'While' '('(logCond)*')' 'Do' block ';'
;
.....
WS
      :     (' ' | '\r'|'\t'|'\u000C'|'\n') {$channel=HIDDEN;}
      ;
COMMENT
      :     '/*' (options {greedy=false;} : . )* '*/'
{$channel=HIDDEN;}
      ;
LINE_COMMENT
      :     '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
      ;
```

Gambar 7. Potongan grammar Algoritmik.g

Tipe grammar yang di pilih dalam penelitian ini adalah gabungan parser dan lexer dengan output template, semua rule yang di tulis juga dapat di visualisasikan dalam syntax diagram, yang akan di pakai dalam editor translator, gambar 8 berikut merupakan salah satu contoh syntax diagram dalam rule statemen,



Gambar 8. Syntax diagram rule statement

Sementara itu string template juga di tulis dengan tangan (hand code) dan di beri nama file *algoritmik.stg* dan potongan filenya seperti terlihat pada gambar 9 berikut,

```
group Algoritmik;
program(libs,globals,functions,mainfunctions) ::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
/*library*/
setLib(lib)::="#include\<<setFile(lib)>\>"
setFile(name)::="<name>"
getTypedeclarator(list)::= <<<list;separator=",">
>>
/*main program*/ //int argc,char *argv[]
mainfunct(arg,locals,stats)::=<<int main(){
  <locals; separator="\n">
  <stats; separator="\n">
  return 0;
}
>>
....
.....
```

Gambar 9. String Template Notasi Algoritmik

Grammar dan string template di atas merupakan kernel dari translator notasi algoritmik yang berfungsi mengenali input (source code) sesuai grammar (lexer dan parser) sehingga setelah valid akan di translasikan sesuai string template yang di buat, kernel lain yang di pakai adalah library ANTLR yang di integrasikan dalam class tersendiri yang di tulis dalam java. Model grammar menyesuaikan dengan model notasi algoritmik yang di pilih untuk penelitian ini (Liem, 2007, Wijanarto, 2012) dengan beberapa modifikasi yang bertujuan untuk memudahkan implementasi model notasi algoritmik ke dalam grammar.

5.1.1.b. Implementasi RAD dalam kerangka MVC

Hasil implementasi metode RAD dalam kerangka MVC di tulis dengan paradigma object oriented dan akan di paparkan pada bagian di bawah ini.

A. Fase Planning

B.1 Tujuan

Menghasilkan aplikasi atau tool (*editor teks*) yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C. Sehingga alat ini akan membantu mahasiswa atau orang yang tertarik di bidang pemrograman untuk fokus belajar bagaimana menyelesaikan masalah dengan notasi algoritmik standar yang sederhana tanpa memikirkan bahasa pemrograman yang terkesan rumit dan membingungkan.

B.2 Fungsional

Fungsi sistem ini adalah sebagai media yang menjembatani pengajar/dosen melakukan pendampingan kepada pembelajar/mahasiswa yang melakukan proses pembelajaran notasi algoritmik. Tabel 7 berikut dapat lebih detail menjelaskan kebutuhan fungsional aplikasi yang di bangun.

Tabel 7 Fungsi dan Fasilitas Translator Notasi Algoritmik

Kode	**Keterangan**
Fungsi Dasar	
D.1	Memberikan fasilitas untuk menulis notasi algoritmik baru
D.2	Memberikan fasilitas untuk membuka file yang menyimpan notasi algoritmik
D.3	Menyediakan fasilitas untuk run program
D.4	Menyediakan fasilitas untuk build & compile

D.5	Menyediakan fasilitas untuk translate notasi algoritmik ke bahasa C
Fungsi Grammar Constructor	
C.1	Menyediakan fasilitas untuk menulis Algorithmic Notation Grammar
C.2	Menyediakan fasilitas untuk menulis String Template for C Language

B.3 Ruang Lingkup

Rancangan sistem yang dibuat merupakan sebuah sistem yang mampu mentranslasikan notasi algoritmik standar dalam bahasa C, berisi modul utama program berupa *text editor* yang menjadi pengantar muka proses pembuatan notasi algoritmik sampai proses translasi menjadi code program standar dalam bahasa C.

Translator Notasi Algoritmik ini memberikan fasilitas untuk menulis notasi algoritmik baru, membuka notasi yang sudah tersimpan, melakukan *run program*, *build & compile* dan fasilitas untuk mentranslasikan notasi algoritmik ke bahasa C.

B. Fase User Design

B.1 Use Case Diagram

Diagram *Use-case* di sini dijabarkan secara grafis yang menggambarkan interaksi antara sistem, sistem eksternal, dan pengguna dari aplikasi Translator Notasi Algoritmik yang dibangun. Diagram *Use-case* berikut menggambarkan siapa saja yang akan menggunakan aplikasi Translator Notasi Algoritmik, dan bagaimana cara pengguna berinteraksi dengan sistem yang dibuat.

Use-case Naratif akan menjelaskan sekuensi langkah-langkah dari setiap interaksi yang terjadi, sesuai dengan metode yang digunakan maka langkah langkah yang dilakukan dalam membuat *use-case* diagram adalah sebagai berikut:

a) **Mengidentifikasikan Pelaku Bisnis**

Disini pelaku bisnis diartikan sebagai pemakai, user atau aktor yang akan terlibat dalam sistem ini, daftar aktor bisa dilihat pada tabel 8 berikut :

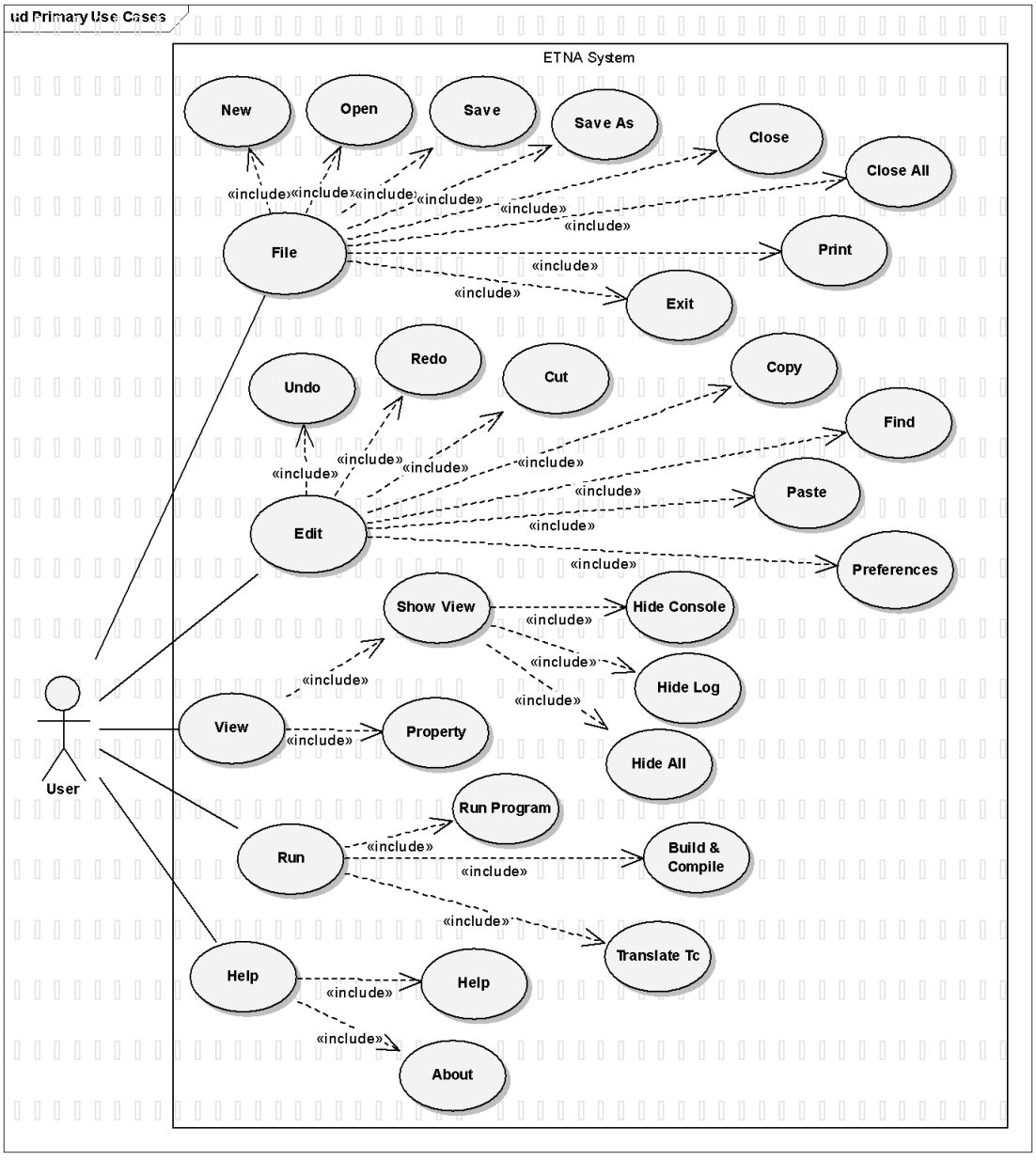
Tabel 8. Identifikasi Pelaku Bisnis

Istilah	Sinonim	Deskripsi
1. Grammar Constructor	Grammar Constructor	Individu atau orang yang membuat Algoritmic Notation Grammar dan membuat String Template for C Language

2. User	Pengguna	Individu atau orang yang memakai program Translator Notasi Algoritmik

b) **Diagram model *Use-case* ETNA System**

Editor translator notasi algoritmik ini akan di beri nama ETNA yang merupakan singkatan dari Editor Translator Notasi Algoritmik, dan gambar 10 di bawah ini merupakan model utama dari system ETNA yang di bangun.

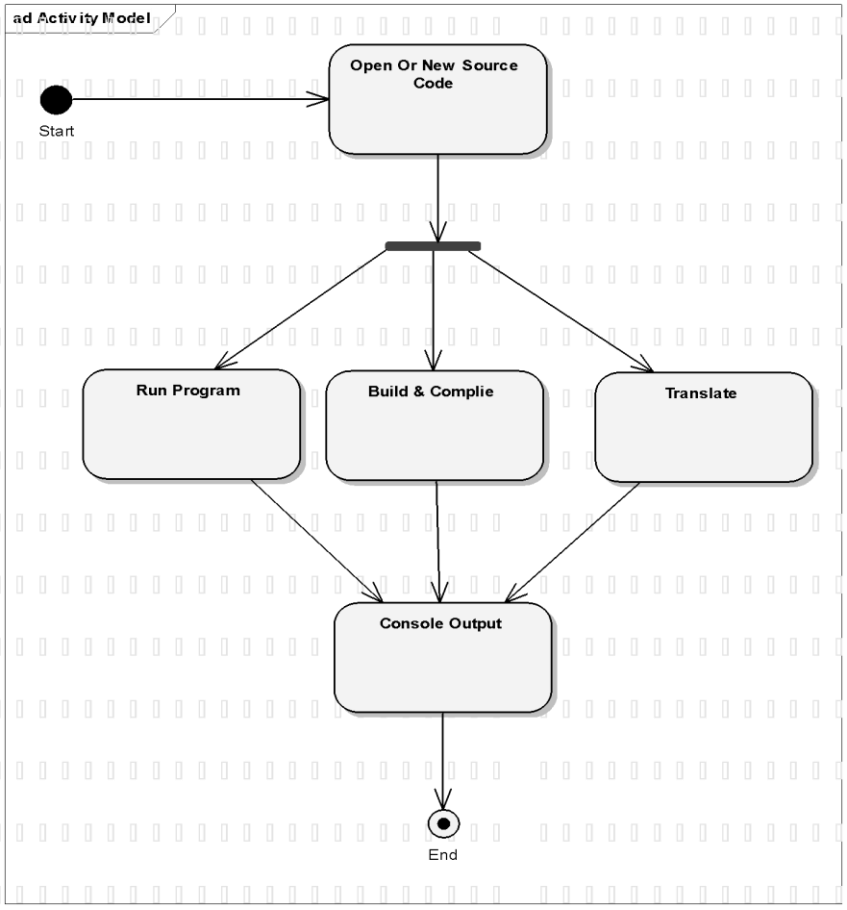


Gambar 10. Model Use case ETNA.

Gambar di atas menjelaskan hubungan antara user/pengguna dengan *use case*. User/pengguna melakukan proses *file, edit, view, run, help*. Dalam proses *file* terdapat fasilitas untuk *new, open, save, save as, close, close all, print* dan *exit*. Dalam proses *edit* terdapat fasilitas untuk *undo, redo, copy, cut, find, paste,* dan *preferences*. Di dalam proses view terdapat fasilitas untuk *show view* dan *property*. Di dalam proses *run* terdapat fasilitas untuk *run program, build & compile* dan *translate To c*. Dan diproses *help* terdapat fasilitas *help* dan *about*.

B.2 Activity Diagram

Diagram aktifitas menggambarkan berbagai alir aktifitas perancangan sistem, bagaimana alir berawal, keputusan yang mungkin terjadi, dan bagaimana berakhir. Grambar 11 di bawah ini menunjukan diagram aktifitas yang di lakukan oleh user saat menggunakan ETNA.

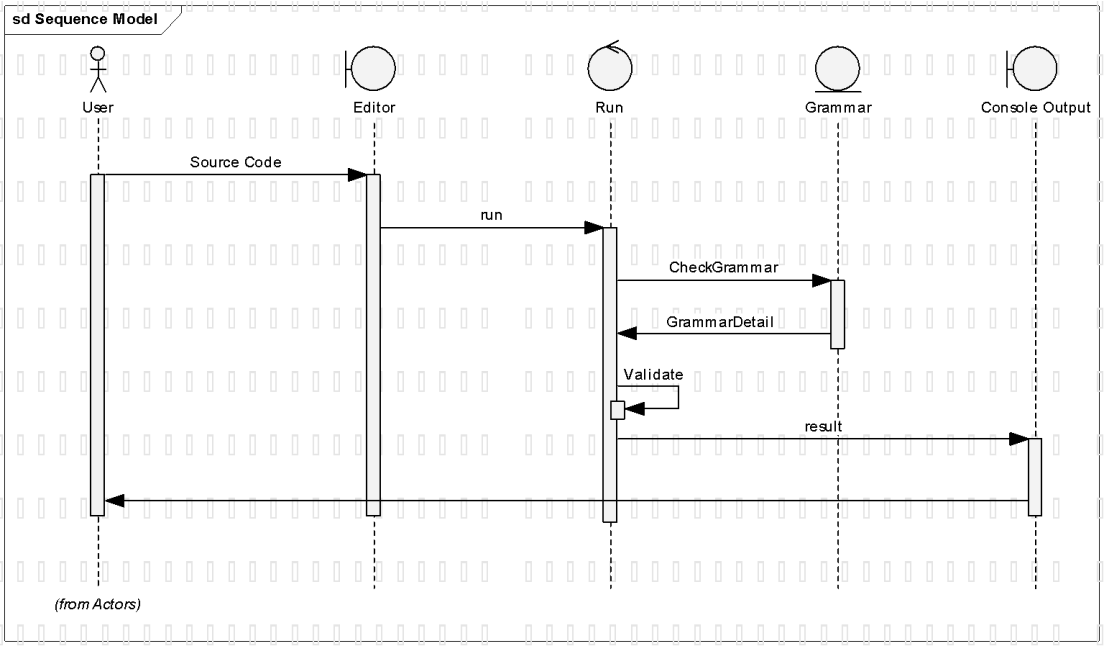


Gambar 11. Diagram Aktifitas ETNA

Gambar 11 di atas menjelaskan pengguna/user akan masuk ke *text editor* kemudian menuliskan notasi algoritmik atau membuka file yang sudah ada, selanjutnya user/pengguna dapat melakukan *run program* atau *build & complie* atau *translate,* hasil dari ke tiga proses tersebut di tampilkan di *console output atau console system*

B.3 Sequence Diagram

Diagram *Sequence* menggambarkan perilaku pada sebuah scenario dapat di lihat pada gambar 12 di bawah ini.

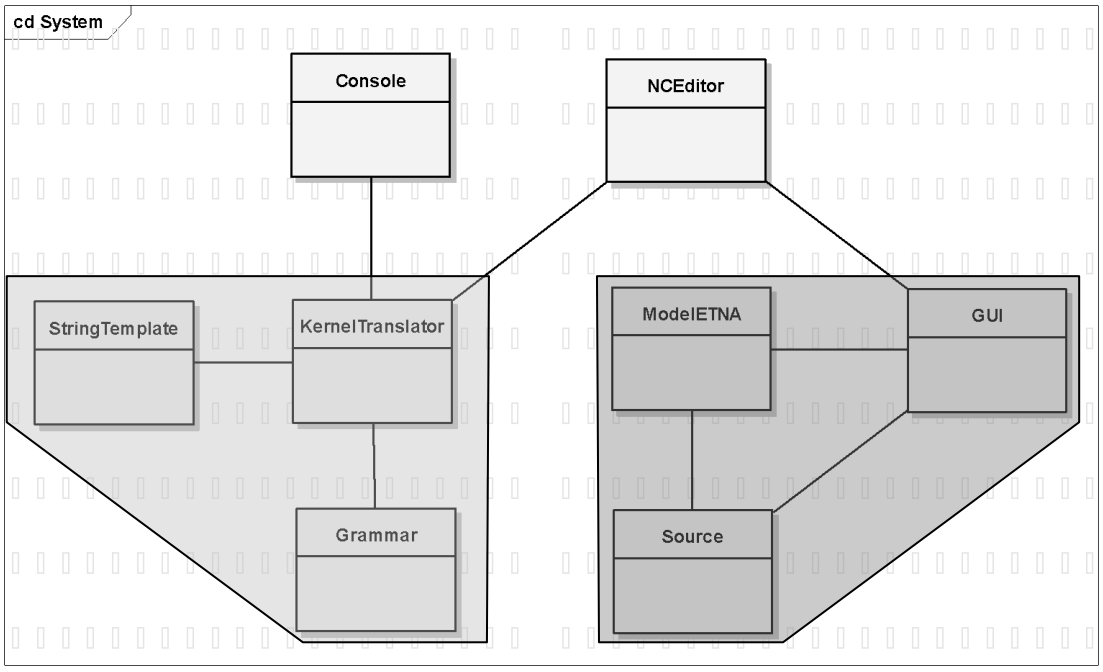


Gambar 12. Diagram Sekuen ETNA

Proses *run program, user* memasukkan notasi algoritmik, notasi algoritmik diverifikasi dengan mengecek *grammar,* bila notasi algoritmik benar atau tidak sesuai dengan *grammar* maka hasil ditampilkan di *console output*.

B.4 Class Diagram

Diagram *class* di sini menggambarkan hubungan antar *class* dan hubungan *class* dengan *class* pendukungnya. Gambar 13 diagram *class* dapat dilihat pada gambar berikut, sebagai catatan tidak semua class di gambarkan dalam laporan ini mengingat class ETNA sangat besar karena melibatkan library dari luar dan yang penting saja yang di tampilkan dalam gambar 13 :

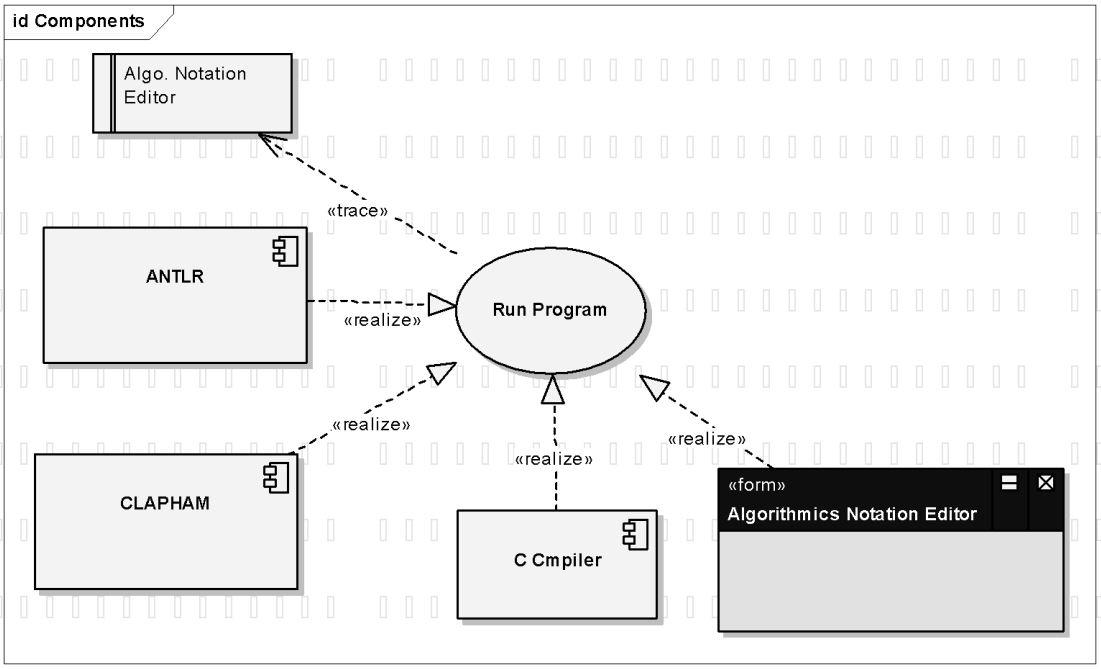


Gambar 13. Class Diagram ETNA

Terlihat pada gambar di atas, spot bidang abu-abu muda di bagian kiri, merupakan kernel utama dari ETNA yang di tulis dengan tangan (hand coded), sementara pada spot bidang abu-abu tua di sebelah kanan merupakan implementasi moel MVC yang di ekspresikan dengan metode RAD.

B.5 Implementasi Diagram

Diagram implementasi menggambarkan hubungan antar komponen yang terlibat dalam sistem. Gambar 14 di bawah menunjukan fungsi utama dalam kernel (menu Run) dalam GUI ETNA serta beberapa tool yang di pakai dalam kernel (ANTLR, C Compiler, Clapham) serta kernel utama (grammar dan string template).

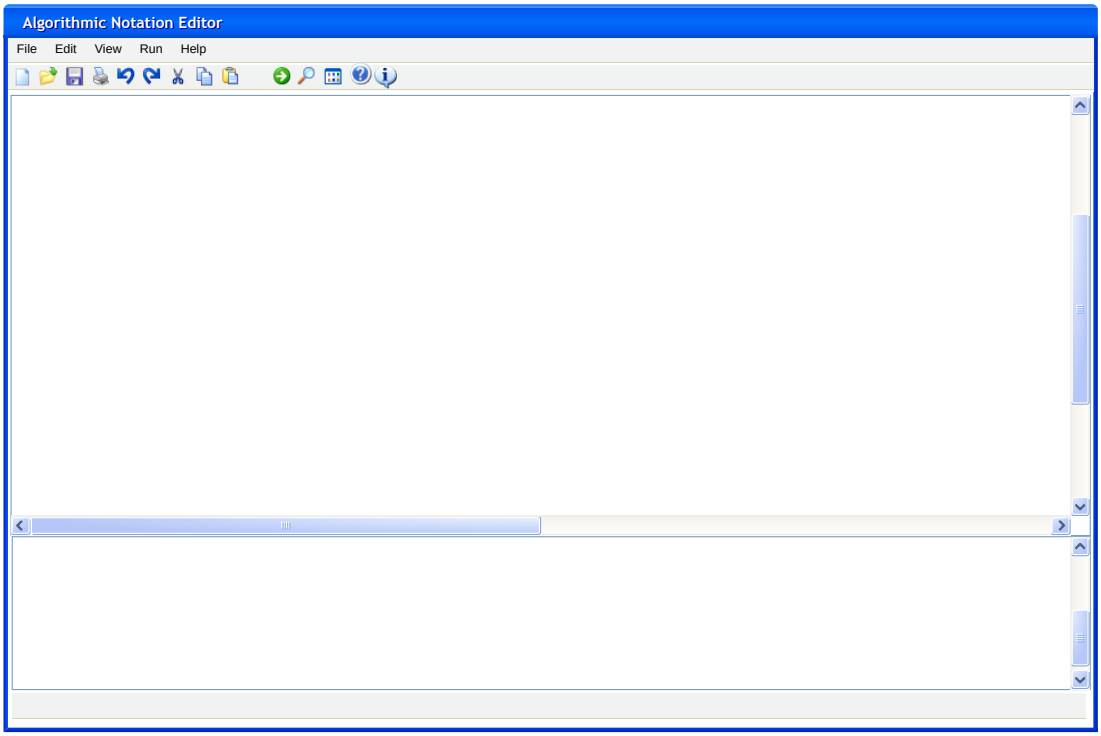


Gambar 14 : Implementation Diagram ETNA System

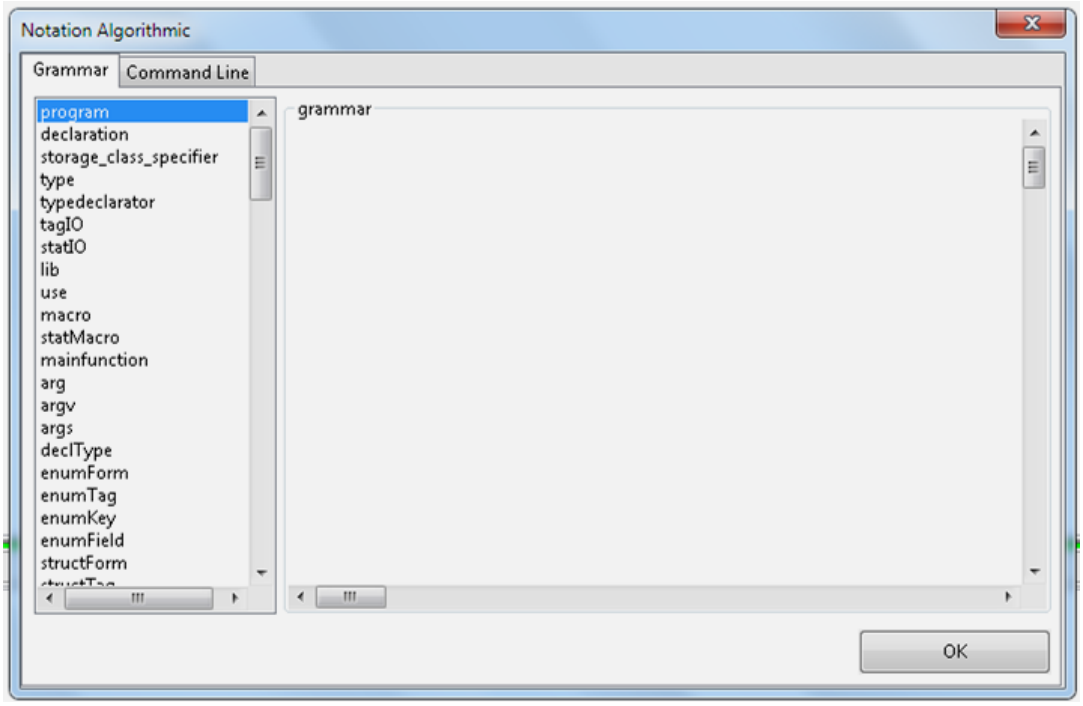
Gambar di atas menjelaskan saat *run* program, sistem memanggil *external tool for parsing input* ANTRL, memanggil *external tool for generate image syntax tree* CLAPHAM, memanggil C *Compiler* yang direalisasikan di *Algorithmic Notation Editor*.

B.6 Disain Interface

Disain interface ETNA, seperti pada editor umumnya, hanya terdiri dari workspace untuk menulis notasi algoritmik, yang di lengkapi dengan menu pendukungnya, serta console output yang di pakai sebaga tempat menampilkan hasil translasi, pesan kesalahan, proses translasi, proses building dan proses running notasi, seperti di tunjukan pada gambar 15 di bawah ini. Tidak semua interface akan di tampilkan dalam laporan ini karena alasan fokus utama interface merupakan workspace untuk menulis notasi, sedangkan interfacea yang lain hanya merupakan mendukung untuk melakukan format editor yang tidak penting untuk di tampilkan dalam penelitian ini. Sementara gambar 15 adalah interface untuk membantu pemahaman user dalam menulis notasi yang di wujudkan dalam syntax diagram tree dari rule yang di buat dalam grammar.



Gambar 15. Disain Interface Utama ETNA



Gambar 16. Disain Bantuan Syntax Tree Untuk User

C. Fase Construction

Pada tahap konstruksi ini rancangan translator notasi algoritmik di implementasikan dalam bentuk editor teks dengan menggunakan bahasa pemrograman *Java SE 1.7 dengan Integrated Deveopment Environtment Eclipse Juno*. Berikut hasil aplikasi ETNA yang di bangun berdasarkan beberapa pendukung baik hardware maupun software sebagai berikut :

Perangkat lunak pendukung yang dibutuhkan adalah :

1. ANTLR 3.4 ke atas
2. Clapham 1.0 ke atas
3. MinGW (GCC Compiler) atau GCC 4.6.2 ke atas
4. Jdk-7-windows-i586 atau Jdk-7-linux32-i586 atau Jdk-7-linux64-i586

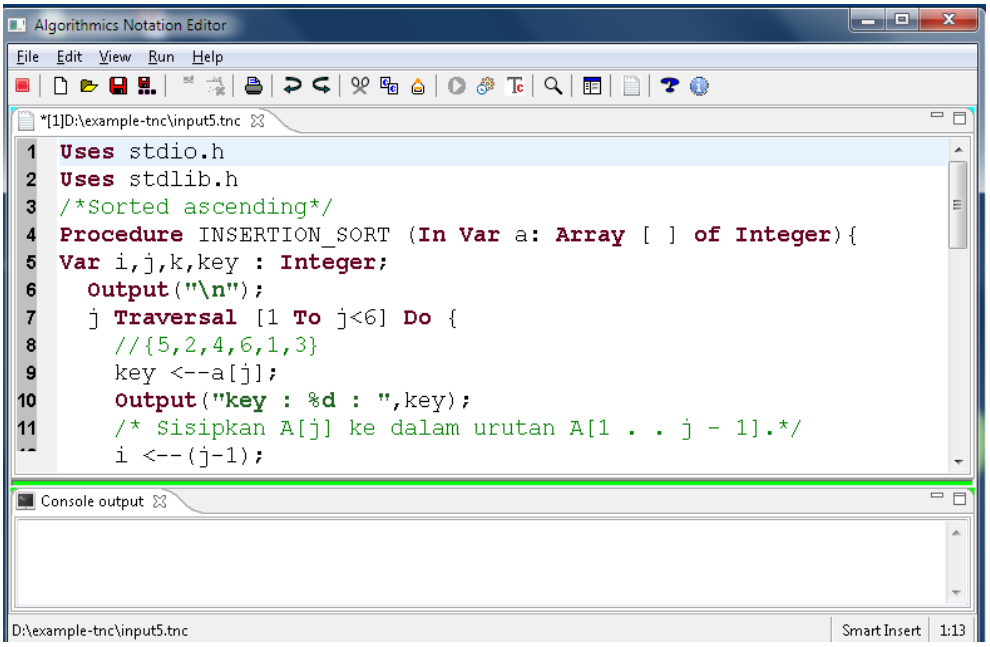
Spesifikasi *hardware* untuk implementasi sistem sebagai berikut :

1. Intel Pentium® Dual-Core
2. RAM 3 GB
3. Resolusi Monitor 1280 x 720

ETNA di disain dapat berjalan pada sistem operasi yang digunakan MS Windows XP Edition SP 3, Windows 7 32 bit, Linux Ubuntu 32 dan 64 bi. Berikut ini hasil implementasi aplikasi ETNA dengan nama file eksekusi Nattoc32Win seperti terlihat pada beberapa gambar 17 berikut :

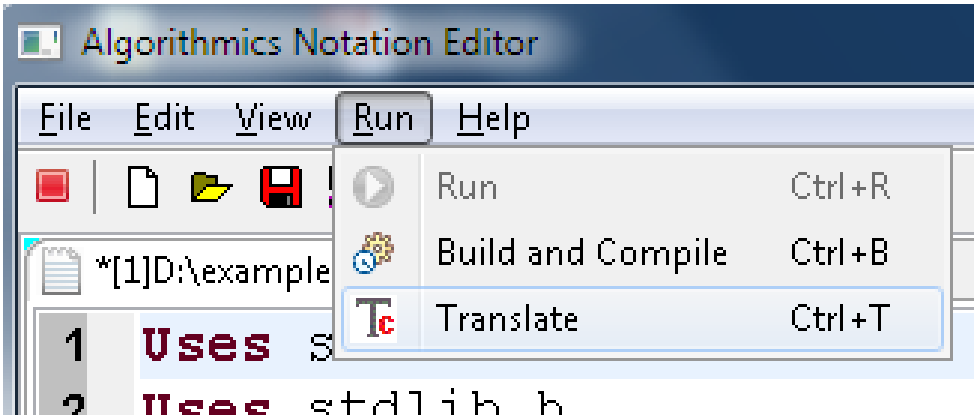
Gambar 17. Start Up Aplikasi ETNA

Gambar 17 merupakan start up aplikasi, sekaligus melakukan inisialisasi terhadap library yang di butuhkan oleh ETNA, termasuk melakukan generate syntax tree untuk bantuan user. Setelah tahap ini selesai maka ETNA siap menerima input (File lama dalam format *.tnc atau user menulis langsung ke editor), gambar 18 menunjukan aplikasi ETNA sedang dalam keadaan menerima file yang sedang terbuka.



Gambar 18. ETNA dengan file notasi aktif

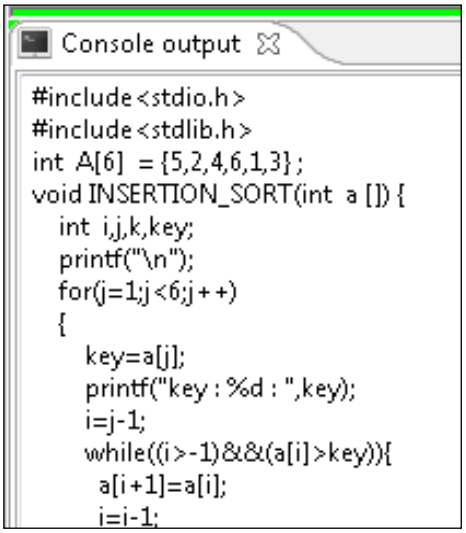
Perlu juga di tampilkan fungsi utama ETNA yang merupakan kernel, yang di implementasikan dalam bentuk menu dalam ETNA seperti terlihat pada gambar 19 berikut.



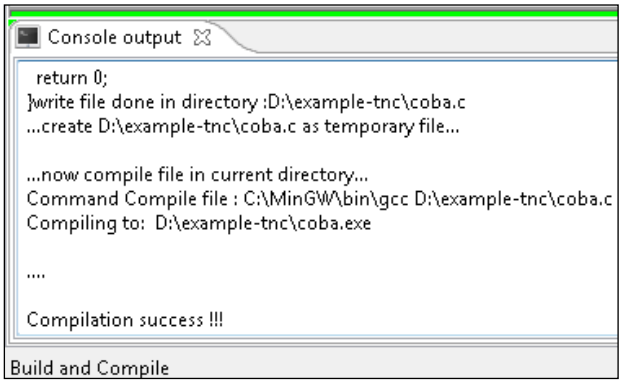
Gambar 19. Menu Utama ETNA

Kernel utama ETNA terdiri dari translate, build and compile serta run, dimana urutan tersebut mennjukan hirarki eksekusi, notasi harus di translasikan lebih dahulu sebelum dapat di build and compile dan setelah dilakukan kompilasi, maka menu run dapat di jalankan untuk memeriksa apakah notasi yang terkompilasi sudah sesuai yang di inginkan.

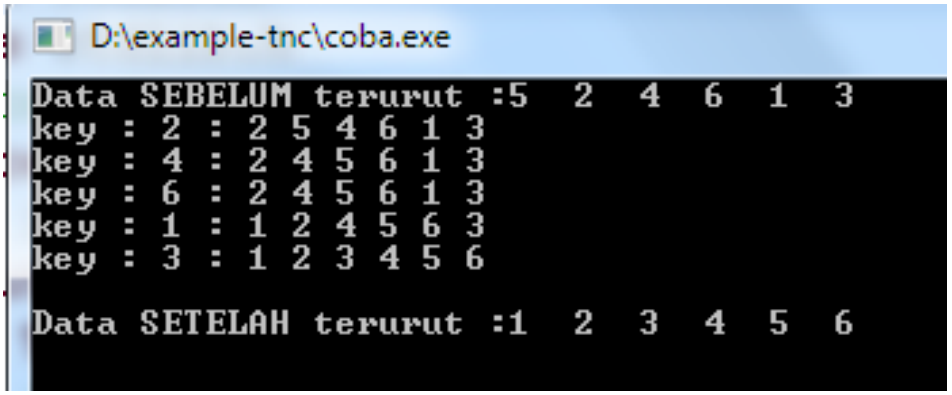
ETNA juga dapat memberikan output berupa hasil translasi, proses build adn compile serta hasil running program baik dalam console aplikasi atau console sistem seperti terlihat berturut -turut a, b dan c dalam gambar 20 berikut,



(a)



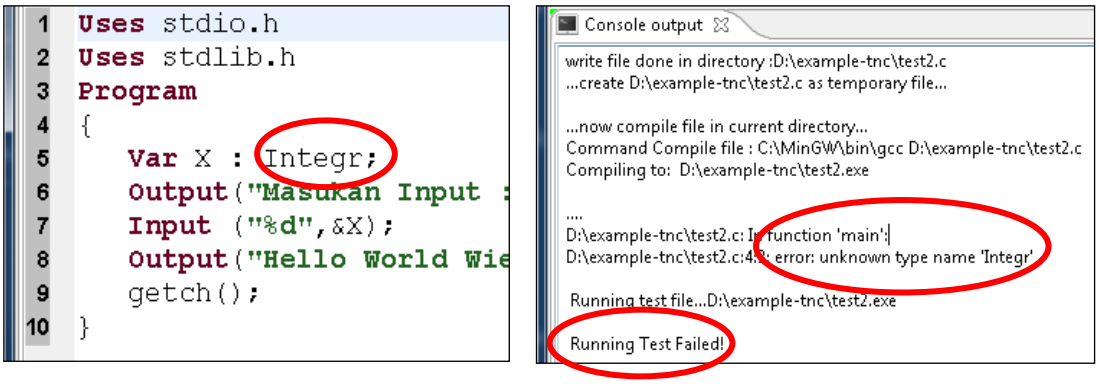
(b)



(c)

Gambar 20. Output ETNA (a) hasil translas, (b) hasil kompilasi (c)hasil running

Output console ETNA juga dapat menampilkan kesalahan yang terjadi jika penulisan notasi salah maka saat di jalankan akan salah pula, seperti terlihat pada gambar 21 di bawah ini.



(a) (b)

Gambar 21. Output ETNA (a) kesalahan notasi (b) hasil kesalahan

D. Fase Cutover

Pada tahap ini hasil aplikasi Translator Notasi Algoritmik di uji coba ke pengguna/user untuk mengetahui dan mengevaluasi hasil aplikasi apakah sudah sesuai dengan rancangan dan tujuan dari pembuatan aplikasi. Pada tahap ini akan di lakukan uji hipotesa dengan analisa kuantitatif yang akan di paparkan pada bagian lain dari laporan penelitian ini.

5.1.2. Hasil Penelitian Eksperimen

5.1.2.1. Uji Persyaratan Analisis

A. Hasil Uji Normalitas Sebaran Data Kelompok Kontrol dan Kelompok Eksperimen

Dari data uji normalitas sebaran data ini diperoleh dari pascates kemampuan memecahkan masalah pemrograman dasar. Menggunakan bantun SPSS 20 dihasilkan nilai *Sig. (2-tailed)* pada *Kolmogorov-Smirnov* yang menunjukkan sebaran data berdistribusi normal apabila nilai *Sig. (2-tailed)* yang diperoleh dari hasil perhitungan lebih besar dari tingkat Alpha 5% (*Sig. (2-tailed)* > 0,050). Hasil uji normalitas sebaran data pascates kemampuan menyelesaikan masalah perograman dasar kelompok kontrol dan kelompok eksperimen selengkapnya dapat dilihat pada lampiran. Sementara itu tabel 9 dan 10 di bawah ini menyajikan ringkasan dari uji statistik dengan SPSS 20, seperti di bawah ini, hasil lengkap di lampiran 9 - 13.

Tabel 9. Rangkuman Hasil Uji Normalitas Sebaran Data Pascates Kemampuan Menyelesaikan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen

Data	Sig.	Keterangan
Pascates Kelompok Kontrol	0.140	Sig. (2-tailed) > 0, 050 = normal
Pascates Kelompok Eksperimen	0.690	Sig. (2-tailed) > 0, 050 = normal

Tabel 10. Rangkuman Hasil Uji Normalitas Sebaran Data Pascates Kecepatan Waktu Menyelesaikan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen

Data	Sig.	Keterangan
Pascates Kelompok Kontrol	0. 398	Sig. (2-tailed) > 0, 050 = normal
Pascates Kelompok Eksperimen	0. 293	Sig. (2-tailed) > 0, 050 = normal

Berdasarkan hasil perhitungan program SPSS 20 dapat diketahui bahwa sebaran data normal. Dari hasil perhitungan normalitas sebaran data pascates kemampuan dan waktu memecahkan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen diketahui bahwa data-data di atas berdistribusi normal. Jadi data tersebut memenuhi syarat untuk dianalisis dengan statistik Uji-t.

5.1.2.2. Hasil Uji Homogenitas Varian

Setelah dilakukan uji normalitas sebaran data, selanjutnya dilakukan uji homogenitas varian. Menggunakan bantuan program SPSS 20 dihasilkan skor yang menunjukkan varian yang homogen adalah apabila signifikansinya lebih besar dari 0,05. Hasil penghitungan uji homogenitas kemampuan mahasiswa dapat dilihat pada tabel 11 berikut,

Tabel 11. Uji Homogenitas Varian Kemampuan Pascates dari Kelompok Kontrol dan Kelompok Eksperimen

Data	f	df	Sig	Keterangan
Pascatest	0.302	74	0.584	Asymp. Sig. (2-tailed) > 0, 050 = homogen

Sedangkan pada kecepatan pengerjaan masalah penghitungan uji homogenitas dapat di lihat pada tabel 12 di bawah ini,

Tabel 12. Uji Homogenitas Varian Kecepatan Pascates dari Kelompok Kontrol dan Kelompok Eksperimen

Data	f	df	Sig	Keterangan
Pascatest	3.177	74	0.079	Asymp. Sig. (2-tailed) > 0, 050 = homogen

Dilihat dari tabel hasil perhitungan uji homogenitas varian di atas, dapat diketahui bahwa data pascates kemampuan dan kecepatan menyelesaikan masalah pemrograman dasar dalam penelitian ini mempunyai varian yang homogen. Hasil uji homogenitas varian dari pascates kemampuan memecahkan masalah pemrograman dasar selengkapnya dapat dilihat pada lampiran.

Hasil perhitungan uji homogenitas varian pascates memecahkan masalah pemrograman dasar dengan program SPSS 20 dalam penelitian ini menunjukkan bahwa data tersebut telah memenuhi syarat untuk dianalisis dengan analisis statistik Uji-t.

5.1.2.3. Deskripsi Data Hasil Penelitian

Penelitian ini bertujuan untuk mengetahui perbedaan kecepatan memecahkan masalah pemrograman dasar mahasiswa antara yang diberi pembelajaran dengan menggunakan media translator notasi algoritmik dan pembelajaran tanpa menggunakan media translator notasi algoritmik. Setelah itu penelitian ini bertujuan untuk menguji kecepatan penggunaan translator notasi algoritmik dalam memecahkan masalah pemrograman dasar mahasiswa tahun pertama teknik informatika fakultas ilmu komputer UDINUS.

Data dalam penelitian ini meliputi data skor tes akhir memecahkan masalah pemrograman dasar. Data skor akhir diperoleh dari hasil pascates kemampuan memecahkan masalah pemrogramn dasar. Hasil penelitian kelompok kontrol dan kelompok eksperimen disajikan sebagai berikut.

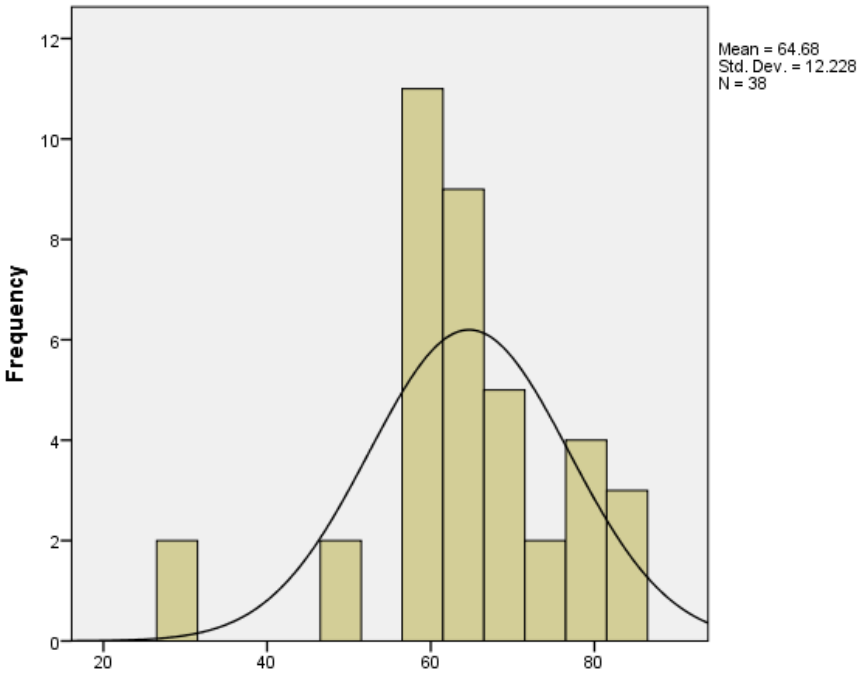
A. Deskripsi Data Pascates Kemampuan dan Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol

Pemberian pascates kemampuan memecahkan masalah pemrograman kelompok kontrol dimaksudkan untuk melihat pencapaian hasil kemampuan memecahkan masalah pemrograman dasar tanpa menggunakan translator notasi algoritmik. Subjek pada pascates kelompok kontrol adalah 38 siswa. Berikut ini sajian distribusi frekuensi skor kemampuan pascates kelompok kontrol.

Tabel 13. Distribusi Frekuensi Perolehan Skor Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol

No	Kelas Interval	Frekuensi	Frekuensi (%)	Frekuensi Kumulatif	Frekuensi Kumulatif (%)
		15	28.3		28.3
1	0 - 49.99	2	3.8	2	32.1
2	50 - 59.99	6	11.3	8	43.4
3	60 - 69.99	18	34.0	20	77.4
4	70 - 84.99	9	17.0	29	94.3
5	85 - 100	3	5.7	32	100.0
Total		53	100.0		

Data di atas menunjukkan hasil pascatest kemampuan memecahkan masalah pemrograman dasar kelompok kontrol memiliki *mean* 64,68, *median* 64,00, *mode 61*, dan SD 12.228 dengan skor tertinggi 85 dan skor terendah 29. Distribusi frekuensi skor pascates kemampuan menyelesaikan masalah pemrorgaman dasar kelompok kontrol dapat dilihat pada histogram berikut ini, lebih lengkap lihat lampiran 9 - 13.



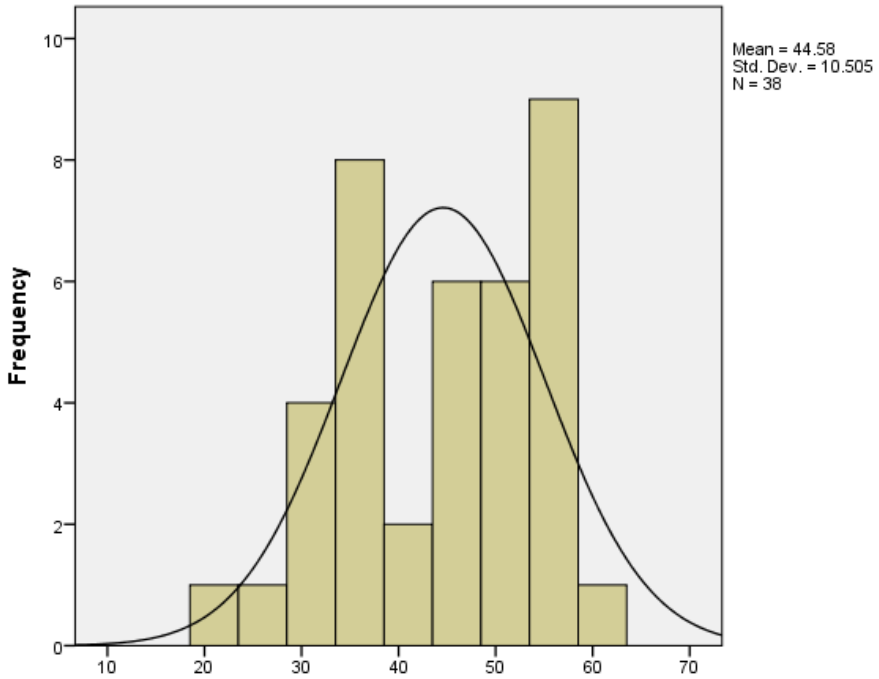
Gambar 22. Histogram Distribusi Frekuensi Skor Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol

Sementara distribusi frekuensi kecepatan menyelesaikan masalah pemrograman dasar di sajikan pada tabel 14 berikut,

Tabel 14. Distribusi Frekuensi Perolehan Skor Pascates Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol

No	Kelas Interval	Frekuensi	Frekuensi (%)	Frekuensi Kumulatif	Frekuensi Kumulatif (%)
		15	28,3		28.3
1	0 – 29.99	3	5,7	3	34
2	30 - 39.99	11	20,8	14	54.8
3	40 - 49.99	10	18,9	24	73.6
4	50 - 59.99	13	24,5	37	98.2
5	60 - 65	1	1,9	38	100.0
Total		53	100		

Data di atas menunjukkan hasil prates kemampuan memecahkan masalah pemrograman dasar kelompok kontrol memiliki *mean* 44,58, *median* 46,33, *mode 56*, dan SD 10.505 dengan skor tertinggi 21 menit dan skor terendah 60 menit. Sedangkan distribusi frekuensi skor pascates kecepatan menyelesaikan masalah pemrograman dasar kelompok kontrol dapat dilihat pada histogram berikut ini.



Gambar 23. Histogram Distribusi Frekuensi Skor Pascates Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol

Melalui histogram di atas dapat diketahui bahwa skor hasil pascatest kemampuan menyelesaikan masalah pemrograman kelompok kontrol didistribusikan menjadi lima kelas interval. Sedangkan keceptan menyelesaikan masalah di di distribusikan menjadi empat kelas interval. Hasil pascates pada kelompok kontrol ini menunjukkan bahwa nilai terbanyak berada pada interval 60 - 69.99, yakni sebanyak 18 akan tetapi rata-rata waktu penyelesaiann masalah berada pada interval 50 - 59.99 sebanyak 13. Hal ini menunjukkan bahwa pada pascates kelompok kontrol, masih banyak siswa yang belum mampu mencapai nilai standar yang berada di kisaran 70 - 84.99 sehingga skor pascates kelompok kontrol masih tergolong agak rendah. Hal ini kemungkinan disebabkan karena mahasiswa belum mempunyai kemampuan menyelesaikkan masalah pemrograman dasar tanpa di notasikan dan penguasaan bahasa program yang rendah, sehingga pascates pada kelompok kontrol hanya meningkat sedikit dan dapat dikatakan kurang memuaskan. Sementara pada kecepatan menyelesaikan masalah pemrograman terlihat bahwa interval yang lebih dari 50 menit berjumlah 13. Sehingga kecepatan waktu menyelesaikan masalah pascatest pada kelompok kontrol masih rendah, kemungkinan di sebabkan kurangnya pemahaman menggunakan bahasa pemrograman secara baik dan benar.

Tabel 15. Kecenderungan Perolehan Skor Pascates Kemampuan Dan Kecepatan Memecahkan Masalah Pemrograman Dasar pada Kelompok Kontrol

No	Kategori	Interval		Frekuensi		Frekuensi (%)	
		A	B	A	B	A	B
2	Rendah	<70	>49	26	14	68.4%	36.84%
3	Sedang	71 s.d 84	30 s.d. 49	9	21	23.7%	55.26%
4	Tinggi	>84	<30	3	3	7.9%	7.90%
			Jumlah	38	38	100%	100%

Keterangan : A adalah Kemampuan dan B adalah Kecepatan

Berdasarkan tabel di atas, dapat diketahui skor kemampuan dan kecepatan dalam kategori rendah yaitu sebesar 68.4% dan 36.84%. Skor kemampuan kategori rendah jika skor berada di bawah 70 yaitu sebanyak 26 siswa dan skor kecepatan di atas 49 ada 14. Sementara itu, skor yang kemampuan dan kecepatan dalam kategori sedang yaitu sebesar 23.7% dan 55.26%. Skor masuk kategori sedang untuk

kemampuan jika skor berada di antara 71 s.d. 84 yaitu sebanyak 9 mahasiswa dan untuk kecepatan 30 s.d. 49 sebanyak 21 mahasiswa. Kemudian skor yang masuk dalam kategori tinggi untuk kemampuan yaitu sebesar 7.9% dan kecepatan tinggi 7.9%. Skor masuk dalam kategori tinggi untuk kemampuan adalah jika skor lebih dari 85 dan kecepatan kurang dari 30.

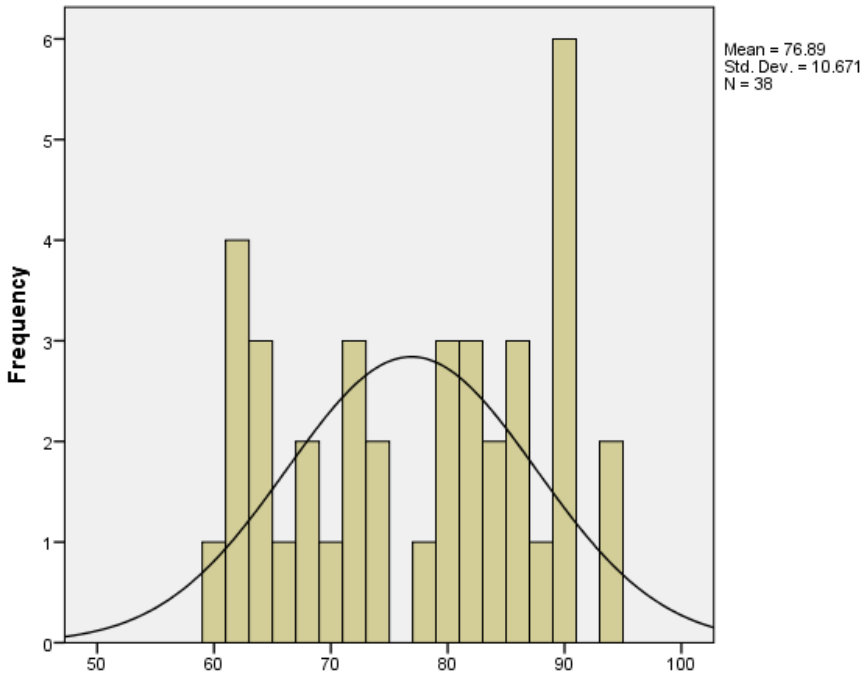
B. Deskripsi Data Pascates Kecepatan dan Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Eksperimen

Pemberian pascates kemampuan dan kecepatan menyelesaikan masalah pemrograman dasar pada kelompok experimen dimaksudkan untuk melihat pencapaian hasil peningkatan kemampuan dan kecepatan memecahkan masalah pemrograman dasar dengan menggunakan translator notasi algoritmik. Subjek pada pascates kelompok experimen adalah 38 siswa. Berikut ini sajian distribusi frekuensi skor kemampuan pascates kelompok eksperimen.

Tabel 16. Distribusi Frekuensi Perolehan Skor Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Experimen

No	Kelas Interval	Frekuensi	Frekuensi (%)	Frekuensi Kumulatif	Frekuensi Kumulatif (%)
		15	28.3	0	28.3
1	0 - 49.99	0	0	15	0
2	50 - 59.99	0	0	15	0
3	60 - 69.99	12	22.6	27	50.9
4	70 - 84.99	14	26.4	41	77.4
5	85 - 100	12	22.6	53	100.0
Total		53	100.0		

Data di atas menunjukkan hasil prates kemampuan memecahkan masalah pemrograman dasar kelompok eksperimen memiliki *mean* 76,89, *median* 79,50, *mode* 61, dan SD 10.671 dengan skor tertinggi 93 dan skor terendah 60. Distribusi frekuensi skor pascates kemampuan menyelesaikan masalah pemrograman dasar kelompok experimen dapat dilihat pada histogram berikut ini.



Gambar 24. Histogram Distribusi Frekuensi Skor Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Experimen

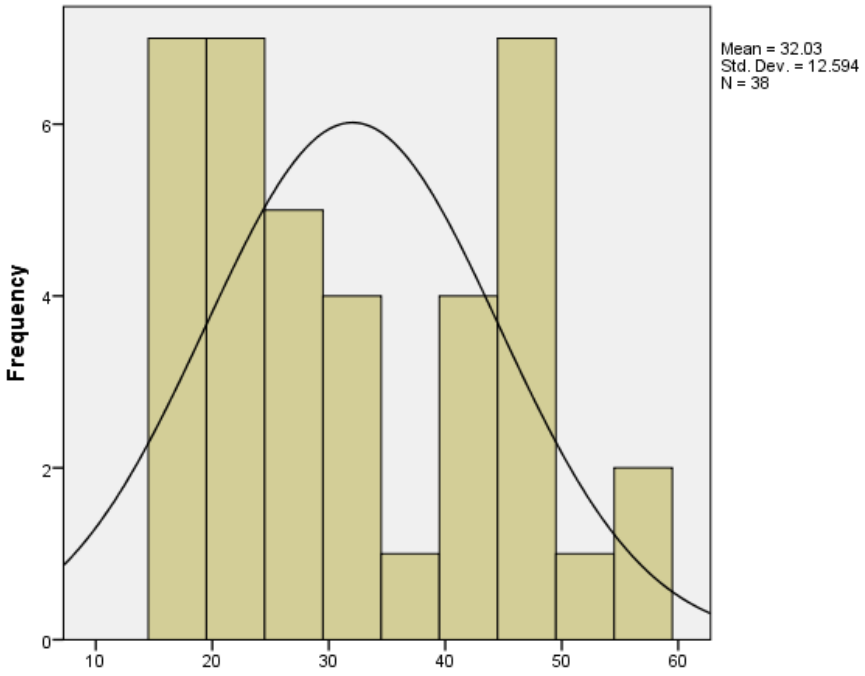
Sementara distribusi frekuensi kecepatan menyelesaikan masalah pemrograman dasar di sajikan pada tabel 17 berikut,

Tabel 17. Distribusi Frekuensi Perolehan Skor Pascates Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Experimen

No	Kelas Interval	Frekuensi	Frekuensi (%)	Frekuensi Kumulatif	Frekuensi Kumulatif (%)
		15	28.3		28.3
1	0 - 29.99	19	35.8	34	64.2
2	30 - 39.99	5	9.4	39	73.6
3	40 - 49.99	11	20.8	50	94.3
4	50 - 50.99	3	5.7	53	100.0
Total		53	100.0		

Data di atas menunjukkan hasil pascatest kemampuan memecahkan masaah pemrograman dasar kelompok eksperimen memiliki *mean* 32,03, *median* 29,00, *mode* 17, dan SD 12.594 dengan skor tertinggi 17 menit dan skor terendah 57 menit.

Sedangkan distribusi frekuensi skor pascates kecepatan menyelesaikan masalah pemrograman dasar kelompok experimen dapat dilihat pada histogram berikut ini.



Gambar 25. Histogram Distribusi Frekuensi Skor Pascates Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Experimen

Melalui histogram di atas dapat diketahui bahwa skor hasil pascatest kemampuan menyelesaikan masalah pemrograman kelompok experimen didistribusikan menjadi lima kelas interval. Sedangkan keceptan menyelesaikan masalah di di distribusikan menjadi empat kelas interval. Hasil pascates pada kelompok experimen ini menunjukkan bahwa nilai terbanyak berada pada interval 70 – 84,99, yakni sebanyak 14. Hal ini menunjukkan bahwa pada pascates kelompok experimen, sudah banyak mahasiswa yang mampu mencapai nilai standar yang di inginkan sehingga skor pascates kelompok experimen sudah tergolong sedang. Hal ini disebabkan karena mahasiswa sudah mempunyai kemampuan menyelesaikkan masalah pemrograman dasar dengan di notasikan secara algoritmik tanpa harus menguasai bahasa program, sehingga pascates pada kelompok experimen cukup meningkat signifikan dan dapat

dikatakan sudah memuaskan. Sementara pada kecepatan menyelesaikan masalah pemrograman terlihat bahwa interval 0 hingga 29,99 berjumlah 19. Sehingga kecepatan waktu menyelesaikan masalah pascatest pada kelompok experimen sudah cepat, kemungkinan di sebabkan penggunaan translator langsung dapat menyelesaikan masalah tanpa memikirkan bahasa pemrograman secara baik dan benar.

Tabel 18. Kecenderungan Perolehan Skor Pascates Kemampuan Dan Kecepatan Memecahkan Masalah Pemrograman Dasar pada Kelompok Experimen

No	Kategori	Interval		Frekuensi		Frekuensi (%)	
		A	B	A	B	A	B
2	Rendah	<70	>49	12	3	31.6%	7.9%
3	Sedang	70 s.d 84	30 s.d. 49	14	16	36.8%	42.1%
4	Tinggi	>84	<30	12	19	31.6%	50%
			Jumlah	38	38	100%	100%

Keterangan : A adalah Kemampuan dan B adalah Kecepatan

Berdasarkan tabel di atas, dapat diketahui skor kemampuan dan kecepatan dalam kategori rendah yaitu sebesar 31,6% dan 7.9%. Skor kemampuan kategori rendah jika skor berada di bawah 70 yaitu sebanyak 12 siswa dan skor kecepatan di atas 49 sebanyak 3. Sementara itu, skor yang kemampuan dan kecepatan dalam kategori sedang yaitu sebesar 36.8% dan 42.1%. Skor masuk kategori sedang untuk kemampuan jika skor berada di antara 70 s.d. 84 yaitu sebanyak 14 mahasiswa dan untuk kecepatan 30 s.d. 49 sebanyak 16 mahasiswa. Kemudian skor yang masuk dalam kategori tinggi untuk kemampuan yaitu sebesar 31.6% dan kecepatan tinggi 50%. Skor masuk dalam kategori tinggi untuk kemampuan adalah jika skor lebih dari 84 dan kecepatan kurang dari 30.

C. Perbandingan Data Kelompok Kontrol dan Kelompok Eksperimen

Berikut ini disajikan tabel perbandingan data pascatest skor tertinggi, skor terendah, *mean*, *median*, dan *mode* dari kelompok kontrol dan kelompok eksperimen. Tabel ini di buat untuk memberi kejelasan gambaran hasil penelitian yang di peroleh sedemikian rupa sehingga kita dapat memperbandingan dengan mudah.

Tabel 19. Perbandingan Data Statistik Pascates Kemampuan dan Kecepatan Menyelesaikan Masaah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen.

Data	N	Score Minimal		Score Maksimal		Mean		Median		Modus	
		A	B	A	B	A	B	A	B	A	B
Pascatest Kelompok Kontrol	38	29	60	85	21	64.68	44.58	64.00	46.33	61	56
Pascatest Kelompok Eksperimen	38	60	57	93	17	76.89	32.03	79.50	29.00	61	17

Keterangan : A: Kemampuan ; B:Kecepatan

Berdasarkan tabel di atas, dapat dilihat bahwa kemampuan memecahkan masalah pemrograman rata-rata atau *mean* pascates pada kelompok kontrol adalah 64.68. Untuk modus dengan skor dari 61 dan median dengan skor 64 Skor tertinggi kemampuan kelompok kontrol sebesar 85 dan skor terendah sebesar 29. Sementara pada kecepatan menyelesaikan masalah pemrograman pascates kelas kontrol skor tertinggi 21 dan skor terendah sebesar 60. Rata-rata atau *mean* kelompok eksperimen sebesar 44.58 dengan median 45.33 dan modus 56.

Sementara itu kemampuan menyelesaikan masalah pada kelompok eksperimen, rata-rata atau *mean* pascates 76.89, median 79.50 dan modus 61. Skor kemampuan terendah di peroleh sebesar 60 dan skor tertinggi adalah 90. Pada kecepatan menyelesaiakan masalah pemrogrmaan di dapat rata-rata 32.03, mena 29.00 dan modus 17, sedangkan skor terlambat (minimal) di peroleh pada kelompok ini adalah 57 dan skor tercepat (maksimal) 17.

Berdasarkan perbedaan besarnya kenaikan nilai rata-rata kelompok kontrol dan kelompok eksperimen maka dapat di katakan terdapat perbedaan antara kelompok pengguna translator notasi algoritmik dan pengguna tanpa translator notasi algoritmik. Namun, perbedaan tersebut perlu diuji signifikansinya agar dapat diketahui perbedaan tersebut signifikan atau tidak, pernghitungan akan dilakukan dengan uji beda menggunakan Uji-t.

5.1.2.4. Hasil Analisis Data untuk Pengujian Hipotesis Pertama

Hasil analisis data untuk pengujian hipotesis yang berbunyi : "Terdapat perbedaan kemampuan menyelesaiakan masalah pemrograman yang signifikan antar kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik" diperoleh hasil perhitungan uji-t yang dilakukan dengan bantuan SPSS 20. Analisis data ini dilakukan untuk menguji hipotesis penelitian yaitu untuk mengetahui perbedaan kemampuan menyelesaikan masalah pemrograman dasar dengan menggunakan translator notasi algoritmik dengan yang tidak menggunakan translator notasi algoritmik.

a. **Hasil Uji-t**

Uji-t dalam penelitian ini digunakan untuk menguji perbedaan kemampuan menyelesaikan masalah pemrograman dasar antara kelompok kontrol yang menggunakan translator notasi algoritmik dan kelompok eksperimen yang menggunakan translator notasi algoritmik. Penghitungan uji-t dilakukan dengan bantuan komputer program SPSS 20. Syarat data bersifat signifikan apabila nilai p lebih kecil dari 0,050.

1) **Uji-t Data Pascates Kemampuan Menyelesaikan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen**

Uji-t data pascates kemampuan menyelesaikan masalah pemrograman dasar kelompok kontrol dan pascates kelompok eksperimen dilakukan untuk mengetahui perbedaan kemampuan menyelesaikan masalah pemrograman mahasiswa kelompok kontrol yang tanpa menggunakan translator notasi algoritmik dan kelompok eksperimen yang menggunakan translator notasi algoritmik. Uji-t selengkapnya dapat dilihat pada lampiran. Berikut adalah rangkuman hasil Uji-t data pascates kemampuan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen .

Tabel 20. Rangkuman Hasil Uji-t Data Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen

Data	T Hitung	df	P	Keterangan
Pascates kelompok kontrol dan kelompok eksperimen	4.638	74	.000	p < 0,05 = signifikan

Tabel 20 menunjukkan bahwa perhitungan menggunakan rumus statistik dengan bantuan komputer program SPSS 20 diperoleh nilai t hitung = 4.638 dengan df = 74, pada taraf signfikansi 0,05% (5%). Hasil Uji-t pascates kemampuan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan nilai p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 (p = 0,000 < 0,05).

Hasil Uji-t tersebut menunjukkan bahwa kedua data memiliki hasil yang berbeda dan terdapat perbedaan yang signifikan. Dengan demikian, hasil Uji-t menunjukkan bahwa terdapat perbedaan kemampuan menyelesaikan masalah pemrograman dasar antara kelompok kontrol yang tidak menggunakan translator notasi algoritmik dan kelompok eksperimen yang menggunakan translator notasi algoritmik.

b. **Pengajuan Hipotesis**

Setelah dilakukan analisis data menggunakan Uji-t, kemudian dilakukan pengujian hipotesis kemampuan menyelesaikan masalah pemrograman dasar. Dengan melihat hasil dari Uji-t, maka dapat diketahui hasil pengajuan hipotesis kemampuan menyelesaikan masalah pemrograman dasar, yaitu hipotesis nol (Ho) yang berbunyi; "Tidak ada perbedaan penggunaan translator notasi algoritmik yang signifikan antara kelompok yang menyelesaikan masalah pemrograman dengan menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik" ditolak.

Sementara itu, hipotesis alternatif (Ha) yang berbunyi; "Terdapat perbedaan kemampuan menyelesaiakan masalah pemrograman yang signifikan antar kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik" diterima. Dengan demikian, dapat disimpulkan bahwa terdapat perbedaan kemampuan menyelesaikan masalah pemrograman dasar pada mahasiswa kelompok kontrol yang tidak menggunakan translator notasi algoritmik dan kelompok eksperimen yang menggunakan translator notasi algoritmik.

5.1.2.5. Hasil Analisis Data untuk Pengujian Hipotesis Kedua

Hasil analisis data untuk pengujian hipotesis kedua yang berbunyi : "Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrograman dibandingkan tanpa menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar" diperoleh dari uji-t selisih skor dan gain skor yang dibantu program SPSS 20. Perhitungan tersebut dilakukan untuk mengetahui keefektifan penggunaan translator notasi algoritmik dalam pemecahan masalah pemrograman dasar.

1. **Uji-t Data Selisih Skor Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen**

Uji-t data pascates kecepatan dalam memecahkan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen dilakukan untuk mengetahui perbedaan kecepatan memecahkan masalah pemrograman dasar mahasiswa kelompok kontrol yang tidak menggunakan translator notasi algoritmik dan kelompok eksperimen yang menggunakan translator notasi algoritmik. Berikut adalah rangkuman hasil Uji-t data pascates kecepatan memecahkan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen

Tabel 21. Rangkuman Hasil Uji-t Data Skor Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen

Data	T Hitung	df	P	Keterangan
Pascates kelompok kontrol dan kelompok eksperimen	-4.718	74	.000	p > 0,05 = signifikan

Tabel 21 menunjukkan bahwa perhitungan menggunakan rumus statistik dengan bantuan komputer program SPSS diperoleh nilai t hitung = -4.718 dengan df= 74, pada taraf signfikansi 0,05% (5%). Hasil Uji-t efektifitas skor kecepatan memecahakan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan nilai p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 (p = 0,000 > 0,05). Dengan demikian, hasil Uji-t menunjukkan keadaan berbanding terbalik bahwa terdapat perbedaan kecepatan skor memecahkan masalah pemrograman dasar antara kelompok kontrol yang tidak menggunakan

translator notasi algortmik lebih lambat dan kelompok eksperimen yang menggunakan translaltor notasi algoritmik lebih cepat. Hasil data tersebut menunjukkan data yang berbeda dan terdapat perbedaan skor kecepatan yang signifikan antara kedua kelompok tersebut.

2. **Gain Skor**

Gain skor disini dimaksudkan sebagai selisih *mean positif* pascates pada kelompok kontrol dan eksperimen. Gain skor digunakan untuk mengetahui adanya peningkatan atau penurunan skor, untuk mengetahui keefektifan translator notasi algoritmik yang digunakan. Gain skor dari kelompok kontrol sebesar 44.58 dan kelompok eksperimen sebesar 32.03. Melalui gain skor tersebut dapat diketahui bahwa skor pada kelompok eksperimen lebih kecil nilainya yang signifikan dibandingkan dengan kelompok kontrol. Dengan demikian, dapat disimpulkan bahwa penggunaan translator notasi algoritmik efektif digunakan dalam pemecahan masalah pemrograman dasar.

3. **Pengajuan Hipotesis**

Setelah dilakukan analisis data menggunakan Uji-t kecepatan skor dan gain skor, kemudian dilakukan pengujian hipotesis kemampuan memecahkan masalah pemrograman dasar. Dengan melihat hasil analisis Uji-t kenaikan skor dan gain skor, maka dapat diketahui pengajuan hipotesis kemampuan memecahkan masalah pemrograman dasar, yaitu hipotesis nol (Ho) yang berbunyi “Penggunaan translator notasi algoritmik tidak lebih cepat memecahkan masalah pemrograman dibandingkan dengan tanpa menggunakan translator notasi” ditolak. Sementara itu, hipotesis alternatif (Ha) yang berbunyi; “Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrograman dibandingkan tanpa menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar” diterima. Dengan demikian, dapat disimpulkan bahwa kecepatan memecahakan masalah pemrograman dasar oleh mahasiswa menggunakan translator notasi algoritmik lebih efektif dibandingkan yang tidak menggunakan translator notasi algoritmik

5.1.3. Pembahasan Hasil Penelitian Eksperimen

Penelitian yang dilakukan penulis adalah di Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswanoro, Semarang, dengan populasi sebanyak 18 kelas pararel, A11.4101 hingga A11.41018. Sedangkan sampel penelitian yang di ambil kelas A11.4101 dan A11.4110, A11.4101 sebagai kelompok eksperimen yaitu kelompok yang pasca test di perlakukan dengan menggunakan translator notasi algoritmik dan A11.4110 sebagai kelompok kontrol, kelompopk yang tidak di perlakukan saat pasca test dengan translator notasi algoritmik.

5.1.3.1. Deskripsi Kondisi Awal Kemampuan Memecahkan Masalah Pemrograman Dasar Pada Kelompok Eksperimen Dan Kelompok Kontrol

Eksperimen tanpa pretest mengasumsikan bahwa, keadaan awal subye enelitian di anggap sama tanpa di perlakukan, karena model pembelajaran yang di berikan juga sama yaitu menggunakan notasi algoritmik pada saat teori maupun praktikum. Beberapa soal standar yang di berikan baik kepada kelompok kontrol maupun kelompok ekperimen menunjukan betapa mereka mengalami kesulitan saat mengeksresikan notasi algoritmik ke dalam bahasa c, yaitu alat standar yang di pakai selama ini. Kesalaan interpretasi terhadap notasi sering di jumpai, hal ini di perparah pemahaman dan penguasaan mahasiswa terhadap bahasa c itu sendiri sangat lemah. Contoh kesalahan baik pada kelompok kontrol maupun kelompok eksperimen saat mengerjakan masalah pemrograman dasar seperti gambar berikut.

```
fmax(data[]):int{
  int i;
  int tmax=-99999;
  for(i=0;i<N;i++){
    If(tmax<data[i])Then {
      tmax=data[i];
    }
  }
  return tmax;
}
```

a

```
int fmax(int data[]){
  int i;
  int tmax=-99999;
  for(i=0;i<N;i++){
    if(tmax<data[i]){
      tmax=data[i];
    }
  }
  return tmax;
}
```

b

Gambar 26. Pekerjaan yang salah (a) dan Pekerjaan yang benar (b)

Seperti terlihat pada gambar 26 a, spesifikasi dan definisi fungsi mengandun argumen yang salah, karena tidak terdapat tipe data. Pada bagian hasil balik fungsi juga salah penempatan, hasil balik fungsi seharusnya terdapat di bagian paling depan sebelum nama fungsi. Demikian juga pada isi algoritma, yaitu analisa kasus melibatkan kata kunci then, hal ini kesalahan interpretasi dari pembacaan notasi algoritmik, yang seharusnya tidak di sertakan dalam tata cara penulisan bahasa c. Sementara itu gambar 26 b merupakan jawaban yang diinginkan dengan benar, dari masalah pencarian nilai ekstrim masksimal, dengan definisi dan spesifikasi yang tepat, argumen parameter benar, hasil balik fungsi tepat serta analisa kondisi yang berlaku pada bahasa c di tulis dengan benar. Jika Gambar 26 a dilakukan kompilasi maka akan menghasilkan kesalahan dan jika intepretasi mahasiswa tidak berubah maka hal ini akan berulang terus dan menyebabkan waktu penyelesaian maslah menjadi lama. Jelas kemampuan dan kecepatan akan menjadi kendala mahasiswa untuk mengerjakan masalah pemrogrman dasar. Secara umum kesalahan yang di

lakukan baik pada kelompok kontrol dan kelompok eksperimen sedikit berbeda. Karena kelompok ekserimen di perlakukan menggunakan translator notasi algoritmik, kesalahan yang sering terjadi pada kelompok ini lebih pada alur logika dan bukan interpretasi notasi algoritmik, karena mereka memakai ETNA, yang memang mendekati notasi yang di pakai dalam teori dan model pembelajaran. Gambar 27 a dan b, menampilkan kesalahan logika yang sering di lakukan pada kelompok eksperimen sebagai berikut

```
Function fmax(In Var data[]: Array
Of Integer):Integer
{
  Var i:Integer;
  Var tmax:Integer<==0;
  i Iterate(0 to N){
    If(tmax<=data[i])Then {
      Tmax<--data[i];
    }
  }
  -> tmax;
}
```

a

```
int fmax(int data[]){
  int i;
  int tmax=-99999;
  for(i=0;i<N;i++){
    if(tmax<data[i]){
      tmax=data[i];
    }
  }
  return tmax;
}
```

b

Gambar 27. Pekerjaan yang salah (a) dan Pekerjaan yang benar (b)

5.1.3.2. Perbedaan Kemampuan dan Kecepatan Memecahkan Masalah Pemrograman Dasar Pada Kelompok Eksperimen Dan Kelompok Kontrol

Pada kelompok eksperimen yang menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar, sedangkan pada kelompok kontrol tidak, yaitu menggunakan compiler bahasa c biasa. Karena tidak diadakan pretest, dan diasumsikan kemampuan baik keolompok kontrol dan kelompok eksperimen sama, maka untuk mengetahui perbedaan kedua kelompok tersebut langsung di lakukan posttest sebanyak 3 kali, yaitu *pertama*, menyelesaikan masalah nilai ekstrim, masalah ini untuk menguji kemampuan penguasaan teknik ekspresi matematis atau statistik dan penggunaan strutur dasar algoritma, *kedua*, masalah bilangan, pada masalah ini mahasiswa di arahkan untuk cenderung dapat memecahkan masalah dengan berpikir logis algoritmis, dengan memanfaatkan pengetahuan dasar mengolah biilangan bulat positif (desimal) dengan oprator pembagian dan modulo dan *ketiga*, masalah pengurutan, masalah klasik dan standar

yang mencakup kemampuan logika, algoritmis dan struktur dasar. Dari ketiga masalah tersebut dapat di kategorikan, masalah pertama ringan, masalah kedua sedang dan masalah ketiga berat untuk mahasiswa tahun pertama.

Pada perlakuan pertama, semua mahasiswa di beri masalah berupa pertanyaan dengan penjelasan atas pertanyaan tersebut dan solusi apa yang di inginkan dari pertanyaan tersebut. Definisi dan spesifikasi masalah menjadi sangat penting, demikian juga aplikasi dari solusi yang di harapkan juga diberikan contohnya. Sehingga mahasiswa hanya menulis solusi yaitu berupa realisasi dari masalah yang dihadapi. Semua masalah di tulis dalam notasi algoritmik seperti model pengajaran yang di pakai dalam kelas teori. Setelah masalah di bagikan asisten akan menerangkan maksud dari permasalahan dan sousi yang di harapkan, jadi peran asisten meakukan repetisi pemahaman atas masalah yang ada, dan di buka diskusi untuk menyamakan persepsi atas masalah dalam dua kelompok yang di uji. Dengan demikian setelah semua jelas maka waktu di hitung untuk mulai mengerjakan solusi atas masalah yang ada. Berikut ini contoh masalah pertama yang di berikan kepada mahasiswa untuk di kerjakan di labortorium dasar.

Problem Test 1. Harga Ekstrim

Diberikan suatu himpunan data yang tidak terurut, yang terdiri dari bilangan bulat positif yang di simpan dalam ***array data***=(3,7,20,5,15,9,25,6,18,12), sebanyak ***N***=10 buah. Jika terdapat variabel global, ***min, max, sum*** untuk menyimpan harga minimal, maksimal dan jumlahan data , ***sumodd, sumeven, maxodd, maxeven, minodd, mineven*** untuk menyimpan masing-masing jumlahan, nilai maksimal, nilai minimal untuk bilangan genap dan ganjil ***avg, avgodd, avgeven*** untuk menyimpan rata-rata semua data, bilangan genap dan bilangan ganjil, serta ***gmax, gmin, gsum, gsumodd, gsumeven, gmaxodd, gmaxeven, gminodd, gmineven***, yang merupakan variabel global untuk menyimpan hasil perhitungan harga ekstrim dari data. Buatlah program yang terdiri dari beberapa fungsi dan prosedur seperti spesifikasi dan definsisi di bawah ini yang mengolah dan menampilkan data seprti di bawah ini :

```
Statistik sederhana
Data : 3 7 20 5 15 9 25 6 18 12
min:25 max:3 sum:120 sumodd:56 sumeven:64
maxodd:20 maxeven:25 minodd:6 mineven:3
avd:12.000000 avdodd:5.600000 avgeven:6.400000
Fungsi : fmin:3 fmax:25 favg:5.599998
```

Berikut Spesifikasi dan Definisi fungsi dan prosedur yang harus di buat :

```
Function  fmax(In data:Array of Integer)→ Integer
/* fmax akan mengembalikan nilai integer, yaitu harga maksimal
   yang merupakan elemen dari array data
*/
Function  fmin (In data:Array of Integer)→ Integer
/* fmin akan mengembalikan nilai integer, yaitu harga minimal
   yang merupakan elemen dari array data
*/

Function  favg(In data:Array of Integer)→ Real
/* favg akan mengembalikan nilai Real, yaitu harga rata-rata
   yang merupakan dari array data.
*/
Procedure  statistik(In Var data:Array of Integer, Out Var gmax,
                     gmin, gsum, gsumodd, gsumeven, gmaxodd,
                     gmaxeven, gminodd, gmineven : Integer,
                     Var avg, avgodd, avgeven : real);
/* statistik akan mengolah nilai dalam array data dan akan
menyimpan hasil pengolahan yang terdiri dari harga maksimal,
minimal, jumlahan, jumlahan data genap, ganjil, harga maksimal
genap ganjil, harga minimal genap ganjil serta rata-rata seluruh
array data, rata-rata data genap dan ganjil ke dalam variabl
global gmax,gmin, gsum, gsumodd, gsumeven, gmaxodd,gmaxeven,
gminodd, gmineven,avg, avgodd, avgeven
*/
```

Mahasiswa kelompok eksperimen, terlihat cenderung memiliki perhatian yang serius dan fokus, serta antusias yang tinggi saat berlangsung perlakuan pascatest 1 ini, Situasi kelas laboratorium menggambarkan keadaan tersebut seperti pada gambar 28 di bawah ini,

Gambar 28. Keadaan Serius dan Fokus pada Kelompok Eksperimen

Perlakuan pascatest 2 pada kelompok eksperiman juga tetap seperti perlakuan pertama, tetap fokus dan antusias seperti tampak pada gambar 29.

Gambar 29. Keadaan Tenang dan Fokus pada Kelompok Eksperimen

Sementara itu pada kelompok kontrol, mahasiswa tampak panik dan tidak percaya diri, terlihat pada gambar 30 a dan b di bawah ini situasi kelas tidak seperti kelompok eksperimen, cenderung saling bertanya pada teman sebelah dan berjalan mencari informasi atas masalah yang di berikan. Bahkan setelah di berikan penjelasan oleh asisten, tetap saja saat mengerjakan masih ada yang bertanya.

a b

Gambar 30 Situasi Kelompok Kontrol Pascatest 1 dan 2 Cenderung Panik dan Tidak Percaya Diri

Pada perlakuan 1 dan 2, baik kelompok eksperimen dan kelompok kontrol masih terdapat kesalahan solusi, namun hasil pengamatan dan evaluasi menunjukan bahwa kelompok eksperimen tidak mengalami masalah pemahaman notasi dan interpretasi, karena cenderung terdapat kesalhan logika, dan ini umum terjadi pada mahasiswa tahun pertama. Sementara kelompok kontrol kesulitan menginterpretasikan dan kurangnya pemahaman akan bahasa c, menghasilkan kesalahan dalam syntax dan logika, sehingga hal ini menghambat penyelesaian masalah dengan durasi waktu yang panjang.

Pada perlakuan 3 untuk soal yang berat, kelompok eksperimen yang sudah terbiasa menggunakan translator notasi algoritmik dalam mengerjakan maslah pemrograman terlihat tenang dan tetap fokus, sementara kelompok kontrol selalu terlihat panik dan tidak fokus, seperti perlakuan 1 dan 2. Gambar 31 menampilkan situasi perlakuan kelompok eksperimen saat mengerjakan masalah pemrograman dasar dengan tingkat kesulitan yang tinggi.

Gambar 31. Situasi Kelas Pada pascatest 3 Kelompok Eksperimen

Uji-t yang untuk mengetahui perbedaan kemampuan menyelesaikan masalah pemrograman dasar antara kelompok yang menggunakan translator notasi algoritmik dengan kelompok yang tidak menggunakan ini dilakukan sebanyak tiga kali. *Pertama*, Uji-t kemampuan menyelesakan masalah pemrograman dasar degan bobot soal yang mudah pada kelompok kontrol dan kelompok eksperimen. *Kedua*, Uji-t pascates kemampuan menyelesakan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen dengan bobot soal yang sedang. *Ketiga*, Uji-t kenaikan skor kemampuan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen dengan soal yang sulit.

Uji-t pascates kemampuan menyelesakan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen dilakukan untuk mengetahui perbedaan kemampuan menyelesakan masalah pemrograman dasar mahasiswa sesudah diberi perlakuan. Hasil perhitungan menunjukkan bahwa besarnya t hitung adalah 4,638 dengan df =74 diperoleh p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 ($p = 0{,}000 < 0{,}05$). Dengan demikian, hasil Uji-t tersebut menunjukkan bahwa

pascates kemampuan menyelesakan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan data yang berbeda dan terdapat perbedaan yang signifikan.

Selisih rata-rata data kenaikan skor kemampuan menyelesakan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen dilakukan untuk mengetahui kenaikan skor kemampuan menyelesakan masalah pemrograman dasar antara kelompok eksperimen yang menggunakan translator notasi algoritmik dengan kelompok kontrol yang tidak menggunakan translator notasi algoritmik. Hasil perhitungan menunjukkan bahwa rata-rata skor kemmpuan pada kelompok kontrol adalah 64.68 sedangkan pada kelompok eksperimen adalah 76.89. Dengan demikian, hasul terdapat selisih rata-rata sebesar, 12.21 yang menunjukkan adanya kenaikan skor kemampuan menyelesakan masalah pemrograman dasar yang signifikan antara kelompok eksperimen yang menggunakan translator notasi algoritmik dan kelompok kontrol yang tidak menggunakan translator notasi algoritmik. Hal ini menunjukkan bahwa setelah diberi perlakuan dengan translator notasi algoritmik kelompok eksperimen lebih mengalami kenaikan jika dibandingkan dengan kelompok kontrol.
Peningkatan kemampuan menyelesakan masalah pemrogrman dasar pada kelompok eksperimen ditunjukkan jawaban mereka berdasarkan parameter yang telah di buat, yaitu prototype, algoritmik, validitas hasil balik fungsi atau prosedur.

Pada prototype penulisan nama fungsi atau prosedur, penentuan argumen sudah terlihat lebih baik, penentuan nama fungsi/prosedur pun sudah tepat. Penentuan pemakaian struktur dasar algoritma menjadi isu penting bagi kelompok komtrol, karena mereka harus merubah nari notasi algoritmik menjadi bahasa formal. Sementara itu, validitas output yang di harapkan juga terihat cukup tepat, hanya saja seringkali penulisan hasil balik fungsi dan nilai hasil balik yang di kirimkan oleh fungsi sering berbeda. Peningkatan mahasiswa yang sangat menonjol terlihat pada pemakaian struktur dasar algoritmik, karena pada kelompok eksperiman hanya di perlukan sedikit perubahan, sementara pada kelompok kontrol sangan berbeda dengan bahasa yang di pakai dalam menyelesaikan masalah.

5.1.3.3. Efektivitas Penggunaan Translator Notasi Algoritmik Dalam Mengerjakan Masalah Pemrograman Dasar Pada Kelompok Eksperimen Dan Kelompok Kontrol

Keefektifan penggunaan translator notasi algoritmik dalam pengajaran pemrograman dasar diketahui dengan analisis uji-t kenaikan skor dan perhitungan gain skor. Perhitungan menggunakan rumus statistik dengan bantuan komputer program SPSS 20 diperoleh nilai t hitung = -4,718 dengan df 74, pada taraf signfikansi 0,05% (5%). Hasil Uji-t kenaikan skor kecepatan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan nilai p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 ($p = 0,000 < 0,05$). Dengan demikian, hasil Uji-t menunjukkan bahwa terdapat kenaikan skor kecepatan menyelesaikan masalah pemrograman dasar antara kelompok kontrol yang tidak menggunakan translator notasi algoritmik dan kelompok eksperimen menggunakan translator notasi algoritmik. Perhitungan selanjutnya menggunakan gain skor. Gain skor disini merupakan selisih *mean* pascates dari masing-masing kelompok kontrol dan eksperimen. Gain skor digunakan untuk mengetui adanya peningkatan atau penurunan skor *mean* masing-masing kelompok. Gain skor kelompok kontrol yakni 44.58, sementara gain skor kelompok eksperimen yakni 32.03. Hasil gain skor tersebut menyatakan bahwa gain skor kelompok eksperimen lebih kecil dari gain skor kelompok kontrol, sehingga dapat disimpulkan bahwa penggunaan translator notasi algoritmik pada kelompok eksperimen lebih efektif dibandingkan dengan kelompok kontrol yang translator notasi algoritmik.

BAB 6. KESIMPULAN DAN SARAN

Setelah melakukan penelitian perekayasaan guna membuat suatu aplikasi, maka hasil sementara yang dapat di simpulkan dalam penelitian ini adalah :

1. Pemilihan model notasi algoritmik dapat di implementasikan ke dalam bentuk grammar yang mudah di implementasikan menjadi suatu aplikasi.
2. Aplikasi yang di bangun merupakan hasil implementasi dari model notasi dan grammar yang disesuaikan berhasil di kembangkan menjadi bentuk editor notasi algoritmik yang di beri nama ETNA (Editor Translator Notasi Algoritmik).
3. Hasil pengujian aplikasi, menunjukan bahwa ETNA berhasil melakukan translasi notasi algoritmik menjadi kode bahasa c yang dapat di kompilasi dan di jalankan dengan sempurna.
4. Aplikasi dapat memberikan respon bila terjadi kesalahan penulisan notasi yang salah dan memberi pesan kesalahan pada user.

Kedepan peelitian ini dapat di kembangkan untuk di beri fitur tambahan dan kemampuan yang lebih cerdas sebagai berikut :

1. Editor di lengkapi dengan teknik *error detection* yang *real time*, artinya jika terdapat kesalahan pada format penulisan notasi oleh user, ETNA di harapkan mampu mendeteksi secara dini dan member pesan kesalahan saat berpindah baris.
2. Hasil deteksi kesalahan dapat di visualisasikan dalam bentuk ekuivalensi kesalahan penulisan notasi dengan syntax tree diagram yang di spesifikasikan oleh grammar yang di pakai.
3. Diharapkan ETNA tidak hanya mampu menjadi translator untuk satu bahasa prosedural saja, tapi dapat menjadi multi translator pada bahasa dengan paradigma prosedural.

6.1 Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dituliskan pada bab sebelumnya dapat disimpulkan sebagai berikut.

1. Terdapat perbedaan yang signifikan antara kemampuan memecahkan masalah pemrograman dasar mahasiswa tahun pertama program studi teknik informatika, FIK Udinus yang menggunakan translator notasi algoritmik dan yang tidak menggunakan translator notasi algoritmik. Hal ini ditunjukkan hasil yang di peroleh pada kecepatan dan kemampuan pada kelompok eksperimen. Mahasiswa kelompok eksperimen lebih menunjukkan kemampuan dan kecetan yang tinggi saat pascatest. Situasi yang berbeda ditunjukkan pada kelompok kontrol. Kelompok kontrol cenderung lamban dalam menyelesaikan masalah karena dalam memecahkan masalah kurang memahami notasi yang di berikan . Perbedaan kemampuan menyelesaikan masalah pada kedua kelompok ini juga telah dibuktikan dengan analisis Uji-t skor pascates antara kelompok kontrol dan kelompok eksperimen yang dilakukan dengan bantuan program SPSS 20 dan dari perhitungan tersebut diperoleh nilai t hitung = 4,638 dengan df 74, pada taraf signfikansi 0,05% (5%). Hasil Uji-t pascates kemampuan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan nilai p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 (p = 0,000 < 0,05).
2. Penggunaan translator notasi algoritmik pada mahasiswa tahun pertama program studi teknik informatika FIK Udinus lebih efektif dibandingkan dengan yang tidak menggunakan translator notasi algoritmik. Hal ini terlihat dari proses menyelesaikan masalah pada kelas eksperimen. Mahaiswa lebih menunjukkan performa yang bagus (cepat). Keefektifan penggunaan translator notasi algoritmik telah dibuktikan dengan analisis uji-t kenaikan skor kelompok kontrol dan kelompok eksperimen dan analisis statistik *gain skor* yang dilakukan dengan bantuan SPSS program 20. Perhitungan Uji-t menggunakan rumus statistik dengan bantuan komputer program SPSS diperoleh nilai t hitung = -4,718 dengan df 74, pada taraf signfikansi 0,05% (5%). Hasil Uji-t kenaikan skor kecepatan menyelesaikan masalah pemrograman dasar kelompok kontrol dan kelompok eksperimen menghasilkan nilai p sebesar 0,000. Nilai p lebih kecil dari taraf signifikansi 0,05 (p = 0,000 < 0,05). Dengan demikian, hasil Uji-t menunjukkan bahwa terdapat kenaikan skor kecepatan menyelesaiakan masalah antara

kelompok kontrol yang menggunakan translator notasi algortmik dan kelompok eksperimen yang tidak menggunakan translator notasi algoritmik. Gain skor dari kelompok kontrol sebesar 44,58 dan kelompok eksperimen sebesar 32,03. Melalui gain skor tersebut dapat diketahui bahwa peningkan skor kecepatan pada kelompok eksperimen lebih mengalami peningkatan yang signifikan.

6.2 Impikasi

Berdasar pada simpulan di atas, beberapa hal yang dapat diimplikasikan dalam penggunaan translator notasi algoritmik yaitu, proses translasi dari masalah ke notasi algoritmik akan berhasil dengan baik jika faktor memahami soal dan menuliskannya ke dalam notasi pemecahan masalah dapat dilakukan dengan tepat. Salah satu faktor pendukung tersebut adalah penggunaan translator notasi algoritmik, karena mahasiswa cukup menulis notasi pemecahan masalah tanpa memikirkan bagaimana bahasa pemrograman yang di pakai. Transalator notasi algoritmik di disain sesuai model pembelajaran pemrograman dasar dari sudut teoritis yang dapat menyajikan penyelesaian yang sesuai dengan materi pemrograman dasar dan bahasa yang di pakai.Penggunaan translator notasi algoritmik mampu menyelesaikan masalah pemrograman yang dapat meningkatkan kecepatan penyelesaian masalah.

6.3 Saran

Berdasarkan simpulan dan implikasi di atas, dapat disarankan beberapa hal sebagai berikut.

1. Dari hasil penelitian ini, terbukti bahwa translator notasi algoritmik efektif dapat digunakan dalam penyelesaian masalah pemrograman dasar pada mahasiswa tahun pertama.
2. Penggunaan translator notasi algoritmik dapat membantu mahasiswa tanpa perlu memikirkan kerumitan bahasa formal yang hanya di ketahui oleh mesin (komuter).
3. Translator notasi algoritmik merupakan alat otomatisasi mahasiswa dalam menyelesaikan masalah pemrograman dasar ke bahasa formal. Selain itu translator notasi algoritmik jelas meningkatkan kemampuan siswa dalam memodelkan masalah ke dalam suatu notasi tanpa perlu memikirkan ekspresinya dalam bentuk bahasa formal.

DAFTAR PUSTAKA

Alfred V Aho, Monica S Lam, Ravi Sethi , Jeffrey D Ullman, 2007, Compilers : principles, techniques, and tools Second Edition. New York : Pearson Education Addison Wesly.

Alverd V Aho, Jeffery D Ullman, 1973, The Theory of Parsing, Translation and Compiling. New York : Prentice Hall Englewood Cliffs, 1973. 0-13-914564-8.

Arikunto, Suharsimi. 2006. Prosedur Penelitian suatu Pendekatan Praktek Edisi Revisi V. Yogyakarta: Rineka Cipta.

Andrew W Appel, Maia Ginsburg, 1998, Modern Compiler Implementation In C. New York : CAMBRIDGE UNIVERSITY PRESS.

Blass, Andreas; Gurevich, Yuri., 2003, Algorithms: A Quest for Absolute Definitions, Bulletin of European Association for Theoretical Computer Science.

Chairmain Cilliers, Andre Calitz, Jean Greyling, 2005, The Application of The Cognitive Dimension Framework for Notations as an Instrument for the Usability analysis of an Introductory Programming tool, Alternation Journal, 12.1b, p 543-576 ISSN 1023-1757.

Chen Shyi-Ming, Lin Chung-Hui, Chen Shi-Jay, 2005, Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques, International Journal of Applied Science and Engineering. 3, 2: 89-100.

David A Watt, Deryck F Brown, 2000, Programming Language Processors in Java, Compiler and Intepreter. New York : Pearson Education, Addison Wesly.

David Harel, Yishai A. Feldman, 2004 , Algorithmics: the spirit of computing, Edition 3, Pearson Education, ISBN 0-321-11784-0.

Hindayati Mustafidah, 2007, Prestasi Belajar Mahasiswa dalam Mata Kuliah Pemrograman Dasar Melalui Pembelajaran Kooperatif Model Jigsaw, Paedagogia, Agustus jilid 10 No 2, hal. 126 – 131.

Ian Somerville, 2011, Software engineering, 9th edition, Pearson Education, Addison-Wesly, Boston, Massachusetts.

Kruskal J. B, Jr., 1956, On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedingsof the American Mathematical Society, 7, pp. 48-50.

Liem, Inggriani, 2007, Draft Diktat Dasar Pemrograman (Bagian Prosedural), ITB , Bandung.

Nasution, 2007, Metode Research, Jakarta, Bumi Aksara.

Nurgiyantoro, Burhan. 2009. *Penilaian dalam Pengajaran Bahasa dan Sastra*. Yogyakarta: BPFE UGM.

Reenskaug, Trygve M.H., 1979, *MODELS - VIEWS - CONTROLLERS.* , XEROX PARC.

Reenskaug, Trygve M.H., 1979, *THING-MODEL-VIEW-EDITOR an Example from a planningsystem.* , Xerox PARC technical note May 1979.

Stanchfield, Scott, Advanced MVC Patterns. JavaDude. [Online] 1996-2009. diakses: 10-10- 2012 http://javadude.com/articles/vaddmvc1/mvc1.htm

Stanchfield, Scott. Applying MVC in VisualAge for Java. JavaDude. [Online] 1996 - 2009. diakses: 10-10-2012. http://javadude.com/articles/vaddmvc2/mvc2.html.

Sugiyono, 2010, Metode Penelitian Kuantitatif, Kualitatif dan R &F, Bandung, Alfabeta.

Sukmadinata, Nana, Saodih, 2007, Metode Penelitian Pendidikan, Bandung, Rosdakarya.

Terence Parr, 2007, The Definitive Guide ANTLR reference, Building Domain-Specific Language. Raleigh, North Carolina Dallas, Texas : The Pragmatic Bookshelf.

Terence Parr, 2010, Language Implementation Patterns Create Your Own Domain-Specific and General Programming Languages. Raleigh, North Carolina Dallas, Texas : The Pragmatic Bookshelf.

Terence Parr, Kathleen Fisher, 2011, LL(*): the foundation of the ANTLR parser generator. s.l. : Vol 11 ACM SIGPLAN Notices - PLDI.

Wikipedia, 2013, diakses 6-01-2013, http://en.wikipedia.org/wiki/Text_editor

Wijanarto, Achmad Wahid Kurniawan, 2012, Model Translator Algoritmik ke Bahasa C, Prosiding Kommit, Komputer dan Sistem Intelijen, Vol 7, 464-472 ISSN 2302-3740.

Yuwono Indro Hatmojo, Sigit Yatmono, 2009, Peningkatan Prestasi Mata Kuliah Komputer Dasar Mahasiswa D3 Teknik Elektro FT UNY Menggunakan Metode Belajar Berbasis Masalah, Jurnal edukasi@Elektro Vol. 5, No.1, Maret, hal. 67 – 78

Lampiran 1. Biodata Peneliti

BIODATA KETUA PENELITI

A. Identitas Diri

1	Nama Lengkap (dengan gelar)	Wijanarto, S.Sos.,M.Kom.
2	Jenis Kelamin	Laki-laki
3	Jabatan Fungsional	Asisten Ahli
4	NIP/NIK/Identitas lainnya	0686.11.2009.354
5	NIDN	0628027003
6	Tempat dan Tanggal Lahir	Yogyakarta, 28-02-1970
7	E-mail	wijanarto@dosen.dinus.ac.id
8	Nomor Telepon/HP	081328635965
9	Alamat Kantor	Jl. Nakulo I 5 – 11 Semarang 50131
10	Nomor Telepon/Faks	024-3520165
11	Lulusan yang Telah Dihasilkan	S1 = 10 Orang
12. Mata Kuliah yg Diampu		1. Dasar Pemrograman
		2. Algoritma Dan Pemrograman
		3. Struktur Data
		4. Strategi Algoritma

B. Riwayat Pendidikan

	S-1	S-2	S-3
Nama Perguruan Tinggi	Universitas Brawijaya	Universitas Gajah Mada	
Bidang Ilmu	Ilmu Administrasi	Ilmu Komputer	
Tahun Masuk-Lulus	1990-1995	2004-2006	
Judul Skripsi/Tesis/Disertasi	Aspek Kultural Jawa Dalam Birokrasi Indonesia 1965-1992	Restorasi Citra Digital Dengan Algoritma Inpainting	
Nama Pembimbing/Promotor	Prof. Drs. Ismani, MPA. Drs. Irwan Noor MA.	Drs. Agus Harjoko, MSc.,Ph.D.	

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir

No	Tahun	Judul Peneitian	Pendanaan	
			Sumber*	Jml (Juta Rp)

D. Pengalaman Pengabdian Kepada Masyarakat dalam 5 Tahun Terakhir

No	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber*	Jml (Juta Rp)
1	2009	Campaign Olimpiade Peserta OSN SMA SEMESTA	SMA Semesta	
2	2010	Diklat Pranata Komputer Kejaksaan Tinggi Jateng	Kejaksaan Tinggi Jateng	

3	2013	Pembnaan OSK SMA 3 Semarang	SMA 3 Semarang	

E. Publikasi Artikel Ilmiah Dalam Jurnal dalam 5 Tahun Terakhir

No	Judul Artikel Ilmiah	Nama Jurnal	Volume/Tahun
1	Restorari Citra Digital Dengan Algoritma Inpainting	Techno-Com	Vol. 8 No.1/ 2009
2	Image Retrieval Berdasarkan Properti Statistik Histogram	Techno-Science	Vol. 38 No.2/2009
3	Vulnerabilitas Program Buffer Overflow	Dian	Vol. 10 No.1/2010
4	Solusi Pencarian N-Puzzle Dengan Langkah Optimal : Suatu Aplikasi Pendekatan Fungsional	Techno-Com	Vol. 10. No.3/2011
5	Simulasi Dan Visualisasi Algoritma Greedy Pemilihan Koin Dalam Bentuk Game	Dian	Vol.11 No.3/2011
6	Perancangan Dan Pembangunan Aplikasi Perangkingan Penerimaan Peserta Didik Smp Hasanuddin 04 Semarang Dengan Promethee Method	Techno-Com	Vol. 11 No. 2/2012
7	Portabilitas Aplikasi Perangkingan Seleksi Penerimaan Siswa Baru Dengan Metode Promethee	Techno-Com	Vol. 11 No.4 2012
8	Model Translator Notasi Algoritmik Ke Bahasa C	KOMMIT Gunadharma	Vol. 7/2012

***F*. Pemakalah Seminar Ilmiah (*Oral Presentation*) dalam 5 Tahun Terakhir**

No	Nama Pertemuan Ilmiah / Seminar	Judul Artikel Ilmiah	Waktu dan Tempat
1	KOMMIT	Model Translator Notasi Algoritmik Ke Bahasa C	8 Oktober 2012, Universitas Gunadarma, Jakarta

G. Karya Buku dalam 5 Tahun Terakhir

No	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1	Teori Pengolahan Citra Digital	2009	255	Andi Offset
2	Strategi Dan Analisis Algoritma	2010	147	Universitas Dian Nuswantoro

H. Perolehan HKI dalam 5–10 Tahun Terakhir

No	Judul/Tema HKI	Tahun	Jenis	Nomor P/ID

I. Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial Lainnya dalam 5 Tahun Terakhir

No	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat Penerapan	Respon Masyarakat

J. Penghargaan dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)

No	Jenis Penghargaan	Institusi Pemberi Penghargaan	Tahun

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidaksesuaian dengan kenyataan, saya sanggup menerima sanksi.

Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Penelitian Dosen Pemula.

.

Semarang, 13-03-2013
Pengusul,

Wijanarto, S.Sos., M.Kom

BIODATA ANGGOTA PENELITI

A. Identitas Diri

1	Nama Lengkap (dengan gelar)	Ajib Susanto, M.Kom.
2	Jenis Kelamin	L
3	Jabatan Fungsional	Asisten Ahli
4	NIP/NIK/Identitas lainnya	-
5	NIDN	0615127404
6	Tempat dan Tanggal Lahir	Bojonegoro, 15-12-1974
7	E-mail	ajibsusanto@gmail.com
8	Nomor Telepon/HP	0818455527
9	Alamat Kantor	Jl. Nakula I 5 – 11 Semarang 50131
10	Nomor Telepon/Faks	024-3520165
11	Lulusan yang Telah Dihasilkan	D3 = 7, S1 = 32 Orang
12. Mata Kuliah yg Diampu		1. Pemrograman Berorientasi Obyek
		2. Pemrograman Web
		3. Pemrograman Client Server
		4. Pemrograman Aplikasi Bisnis

B. Riwayat Pendidikan

	S-1	S-2	S-3
Nama Perguruan Tinggi	Universitas Dian Nuswantoro	Universitas Dian Nuswantoro	
Bidang Ilmu	Teknik Informatika	Teknik Informatika	
Tahun Masuk-Lulus	2002-2004	2005-2008	
Judul Skripsi/Tesis/Disertasi	Pemanfaatan Type Data Bertipe Blob dalam File Binary untuk Pengaksesan File Melalui Streaming SQL pada Server Database	Rekayasa Sistem Pengelolaan Pembelajaran Elektronik Berbasis Web (eLMS)	
Nama Pembimbing/Promotor	Dr-Ing. Vincent Suhartono	Dr.Eng. Yuliman Purwanto, M.Eng	

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir

No	Tahun	Judul Peneitian	Pendanaan	
			Sumber*	Jml (Juta Rp)
1	2011	Rekayasa Model ”Supermuseum” Batik Online Untuk Mengenalkan Keaneka Ragaman Motif Batik Di Indonesia Dalam Upaya Meningkatkan Pemasaran Batik Produk Usaha Kecil Dan Home Industry	Penelitian Strategi Nasional, Dirjen DIKTI Jakarta.	90
2	2010	Perancangan Sistem Informasi Perhitungan	LPP	3,5

		Angka Kredit Dosen	Universitas Dian Nuswantoro	

D. Pengalaman Pengabdian Kepada Masyarakat dalam 5 Tahun Terakhir

No	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber*	Jml (Juta Rp)
1	2008	Pelatihan E-Learning dengan MOODLE bagi Guru SMA Negeri 1 Semarang sebagai Instruktur	SMA Negeri 1 Semarang	-
2	2009	Diklat Pranata Komputer Kejaksaan RI sebagai Instruktur	Kejaksaan Tinggi Jateng	-
3	2010	Pelatihan Aplikasi Perkantoran Open Source PNS Kota Semarang	Universitas Dian Nuswantoro	-
4	2010	Juri Javakanmu ”The art of Java Programming for Education” Tingkat Jateng dan DIY	Universitas Dian Nuswantoro	-
5	2010	Pembuat Soal Komputerisasi dalam Seleksi Pengandaan CPNSD Pemerintah Provinsi dan Kabupaten Kota di Jawa Tengah	Universitas Dian Nuswantoro Semarang	-
6	2010	Diklat Pranata Komputer Kejaksaan RI sebagai Instruktur	Kejaksaan Tinggi Jateng	-
7	2010	Juri pada Lomba Pemilihan Guru Berprestasi Dalam Pembuatan Bahan Ajar Mandiri Berbasis Multimedia	Lembaga Penjaminan Mutu Pendidikan (LPMP) Jawa Tengah	-
8	Tahun pelajaran 2010/2011	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Negeri 11 Semarang SMK Negeri 3 Jepara	-
9	2011	Juri pada Lomba Pemilihan Guru Berprestasi Dalam Pembuatan Bahan Ajar Mandiri Berbasis Multimedia	Lembaga Penjaminan Mutu Pendidikan (LPMP) Jawa Tengah	-
10	Tahun Pelajaran 2011/2012	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Negeri 11 Semarang	-
11	2012	Juri Lomba Pengayaan Sumber Belajar (LPSB) Berbasis Blog Guru Dikdas dan Dikmen Tingkat Provinsi Jawa Tengah	BPITKP Dinas Pendidikan Provinsi Jawa Tengah	-

12	2012	Juri Lomba Multimedia Pembelajaran Guru Tingkat SD/MI, SMP/MTS, SMU/SMK	LPMP Jateng	-
13	Tahun Pelajaran 2012/2013	Exsternal Assesor pada Ujian Praktik Kejuruan Animasi dan Multimedia	SMK Perdana, SMK Robi Rodliyah Semarang	-

E. Publikasi Artikel Ilmiah Dalam Jurnal alam 5 Tahun Terakhir

No	Judul Artikel Ilmiah	Nama Jurnal	Volume/Tahun
1	Rekayasa Sistem Pengelolaan Pembelajaran Elektronik Berbasis Web	Majalah Ilmiah DIAN, Udinus Semarang, ISSN 1412-3088	Vol.9/ No. 2/ Mei 2009
2	Rekayasa E-commerce Berbasis Web pada PT. Samwon Busana Indonesia,	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol.9/ No. 3/ September 2009
3	Perancangan dan Implementasi Sistem Kunci Elektronik pada Locker dengan Media Bluetooth	Jurnal Techno Science, FT UDINUS, ISSN 1978-9793	Vol 3/ No. 2/ Oktober 2009
4	Perancangan dan Implementasi Mobile Siadin (M-Siadin) pada Universitas Dian Nuswantoro Semarang Berbasis J2ME	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol. 10/ No.2/ Mei 2010
5	Rancang Bangun Peta Jalur Fiber Optik di Pt. Indonesia Commets Plus Regional Jawa Tengah dan Daerah Istimewa Yogyakarta secara Online	Jurnal Teknologi Informasi, Techno.COM, ISSN 1412-2693	Vol. 10/No. 4 November /2011
6	Kombinasi Algoritma RSA dan Algoritma Cipher Transposisi untuk Keamanan Database	Jurnal DIAN, Universitas Dian Nuswantoro, ISSN 1412-3088	Vol. 11/No.3/ September 2011
7	Rancang Bangun Aplikasi RMI (Remote Method Invocation) untuk Menghubungkan Sistem Pembayaran Udinus dengan Bank Jateng	Jurnal Teknologi Informasi Techno.COM ISSN 1412-2693	Vol. 11/No.2, Mei 2012
8	Teknik Proteksi SQL Injection dengan Konsep AMNESIA pada Aplikasi Web	Journal of Intellgent Systems and Business Intellegence	Vol. 1, No.2, September 2012

		ISSN 2302-268X	
9	Rancang Bangun Aplikasi Penjadwalan Praktikum di Laboratorium Komputer Universitas Dian Nuswantoro dengan Pendekatan Algoritma Genetika	Majalah Ilmiah DIAN, Universitas Dian Nuswantoro Semarang, ISSN 1412-3088	Vol. 12, No.3, September 2012
10	Rancang Bangun Mobile GIS (Geographic Information System) Pencarian Lokasi ATM BNI Di Semarang Pada Media Ponsel Berbasis Android	SNASTIKOM MEDAN	2013

***F*. Pemakalah Seminar Ilmiah (*Oral Presentation*) dalam 5 Tahun Terakhir**

No	Nama Pertemuan Ilmiah / Seminar	Judul Artikel Ilmiah	Waktu dan Tempat
1	Seminar Jurnal Techno Science	Perancangan dan Implementasi Sistem Kunci Elektronik pada Locker dengan Media Bluetooth	2009 Fak. Teknik Udinus
2	COWISBI Pasca Sarjana Udinus	Teknik Proteksi SQL Injection dengan Konsep AMNESIA pada Aplikasi Web	2012 Pascasarjana Udinus
3	Lomba Mading Digital	Cloud Computing	2013 Udinus

***G.* Karya Buku dalam 5 Tahun Terakhir**

No	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1	Pemrograman Jaringan VB 6.0 & MySQL	2012	162	Widya Karya Semarang

***H.* Perolehan HKI dalam 5–10 Tahun Terakhir**

No	Judul/Tema HKI	Tahun	Jenis	Nomor P/ID

***I.* Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial Lainnya dalam 5 Tahun Terakhir**

No	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat Penerapan	Respon Masyarakat

J. **Penghargaan dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)**

No	Jenis Penghargaan	Institusi Pemberi Penghargaan	Tahun

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidaksesuaian dengan kenyataan, saya sanggup menerima sanksi.

Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Penelitian Dosen Pemula.

.

Semarang, 13-03-2013
Anggota Pengusul,

(Ajib Susanto, M.Kom)

Lampiran 2 Prosiding Artikel Ilmiah

TRANSLATOR NOTASI ALGORITMIK DENGAN LL(*) PARSING DAN STRING TEMPLATE

Wijanarto[1)], Ajib Susanto[2]

[1), 2)] Teknik Informatika UDINUS Semarang
Jl Nakula 5 -11,Semarang 50131
Email : wijanarto.nagan@yahoo.com[1)], ajibsusanto@gmail.com[2)]

Abstrak

Pemrograman dasar merupakan pondasi utama seseorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Kesulitan utama seseorang dalam membuat solusi dalam bentuk bahasa formal merupakan masalah tersendiri, selain pemilihan alat atau aplikasi yang tepat untuk membantunya, bahkan untuk orang dengan latar belakang ilmu komputer. Paper ini mencoba menghasilkan Domain Specific Language (DSL) untuk pengajaran pemrograman dasar dengan grammar LL(*), dalam suatu rancangan aplikasi untuk mempermudah penyelesaian masalah dibidang pengajaran pemrograman dasar berbasis notasi algoritmik. Model notasi algoritmik yang di pilih merupakan model yang sudah pernah diterapkan dan diajarkan di perguruan tinggi. Grammar dihasilkan dengan bantuan ANTLR dan string template, yang di sesuaikan dengan model yang di pilih. Hasil dari penelitian ini berupa Editor Translator Notasi Algoritmik (ETNA), yang diperuntukan bagi mahasiswa di tahun pertama, yang dapat mentranslasikan notasi algoritmik ke bahasa c standar. Alat ini diharapkan membantu seseorang atau mahasiswa untuk dapat mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang dipakai.

Kata kunci: *Translator, Notasi Algoritmik, Pemrograman, domain specific language.*

1. Pendahuluan

Pemrograman dasar merupakan pondasi utama sesorang atau mahasiswa yang ingin belajar membuat program untuk menyelesaikan suatu masalah tertentu. Sesederhana apapun, masalah yang harus di pecahkan harus dilakukan secara terstruktur dan ilmiah. Dalam dunia ilmu komputer atau teknik informatika langkah-langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma [1,4]. Seperti diketahui algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu [3], sepert teori graph dalam menentukan lintasan terpendek [7] dan masih banyak lagi. Dalam studi yang pernah dilakukan di Afrika Selatan [2], keberhasilan pembelajaran pemrograman dasar di pengaruhi oleh, (1) lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman, (2) penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan (3) lingkungan kerja aplikasi harus melindungi mahasiswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Artinya kesulitan utama mempelajari pemrograman di karenakan kesulitan bagaimana memahami semantik dari suatu bahasa pemrograman, seperti di jelaskan dalam [2]. Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti [5], lebih menyoroti metode pembelajaran dari aspek pedagogik, di mana capaian mahasiswa dalam belajar pemrograman dasar sangat rendah di karenakan rendahnya partisipasi, keaktifan dalam berdiskusi dan bertanya serta menjawab pertanyaan dalam kuliah. Sedangkan peneliti lain [16], dalam matakuliah sejenis yaitu komputer dasar, menyimpulkan (masih dari aspek pedagogik) bahwa metode belajar berbasis pada masalah dapat meningkatkan pemahaman materi dan prestasi mahasiswa, namun hanya efektif di lakukan dalam satu siklus saja. Dengan demikian diperlukan model yang dapat menyederhanakan struktur dan semantik instruksi, sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman. Paper ini akan mencoba menghasilkan prototype translator notasi algoritmik ke dalam bahasa C standard untuk pengajaran pemrograman dengan grammar

Domain Spesific Language (DLS), sudah muncul sejak lama [22, 23] juga meta programming [21], konsep ini merupakan perbaruan dari teknik konstruksi kompiler yang secara modern [17, 19, 20, 28]. Parsing dan analisa syntax atau lexical (lexer) [13,18, 24, 25, 26, 27] yang merupakan penentu perluasan ekspresi reguler menjadi pokok perhatian dalam membangun translasi dari kode ke bahasa mesin yang di mengerti komputer. Domain Spesific Language (DSL),

merupakan bahasa pemrograman yang di tujukan untuk keperluan masalah dan solusi yang spesifik. Dalam implementasi DSL paradigma MVC banyak digunakan [12, 23].

Batasan kontekstual dan semantik suatu sintak merupakan aspek dari bahasa pemrogram perlu di tentukan, setelah itu baru kita tentukan apakah bahasa tersebut formal atau informal. Dalam praktik, sintak biasanya menggunakan BNF (Backus-Naur Form) atau Extended BNF, karena kemudahan notasinya [18, 19, 20], yang terdiri dari *himpunan berhingga simbol terminal, simbol non terminal, simbol awal dan aturan produksi* $N ::= \alpha \mid \beta$, dimana **N** adalah simbol *non terminal* , **::=** berarti *terdiri dari* serta α adalah *string terminal* atau *non terminal* yang mungkin kosong serta simbol **|** yang berarti *alternatif*, himpunan tadi di sebut sebagai *context-free grammar*, singkatnya *grammar*.

Parsing LL(*) [12] merupakan perbaikan dari LL(*k*) untuk *k*>1, *lookahead* pada LL(*k*) terbatas pada *k* saja, sedangkan dalam LL(*) dapat mengestimasikan berapa kedalaman *lookahead*. Definisi formal dari LL(*), sebagai berikut [13] grammar *G=(N,T,P,S,Π,M)*, dimana *N* adalah himpunan non terminal simbol atau rule, T adalah himpunan terminal simbol atau token, *P* adalah himpunan produksi, $S \in N$ merupakan start simbol, Π himpunan side effect free predikat semantik, dan *M* adalah himpunan aksi (mutator). Gambar 1 berikut selengkapnya mengenai notasi predikat grammar pada LL(*).

$A \in N$	Nonterminal
$a \in T$	Terminal
$X \in (N \cup T)$	Grammar symbol
$\alpha, \beta, \delta \in X^*$	Sequence of grammar symbols
$u, x, y, w \in T^*$	Sequence of terminals
$w_r \in T^*$	Remaining input terminals
ϵ	Empty string
$\pi \in \Pi$	Predicate in host language
$\mu \in \mathcal{M}$	Action in host language
$\lambda \in (N \cup \Pi \cup \mathcal{M})$	Reduction label
$\vec{\lambda} = \lambda_1..\lambda_n$	Sequence of reduction labels
Production Rules:	
$A \rightarrow \alpha_i$	i^{th} context-free production of A
$A \rightarrow (A'_i)$**=>** α_i	i^{th} production predicated on syntax A'_i
$A \rightarrow \{\pi_i\}$**?** α_i	i^{th} production predicated on semantics
$A \rightarrow \{\mu_i\}$	i^{th} production with mutator

Gambar 1. Notasi Predikat Grammar LL(*)

Produksi adalah di nomori untuk mengekspresikan sebelumnya, sebagai alat mengatasi ambigu. Produksi yang pertama mewakili standar CFG, kedua, menotasikan jembatan *syntatic predicate*, jika simbo A diperluas ke α_i hanya jika input saat ini juga tepat ketemu dengan sintak yang di deskripsikan oleh A'_i. Ketiga menotasikan *semantic predicate*, simbol A di perluas ke α_i hanya jika π_i memenuhi konstruksi state. Bentuk terakhir menotasikan aksi, yaitu pengaplikasian rule state berdasarkan mutator μ_i. Aturan turunan predikat grammar untuk mendukung *syntatic*, *semantic* predikat dan *mutator* serta referensi aturan yang di sajikan pada gambar 2 berikut :

$$Prod\ \frac{A \rightarrow \alpha}{(\mathbb{S}, uA\delta) \Rightarrow (\mathbb{S}, u\alpha\delta)} \qquad Action\ \frac{A \rightarrow \{\mu\}}{(\mathbb{S}, uA\delta) \overset{\mu}{\Rightarrow} (\mu(\mathbb{S}), u\delta)}$$

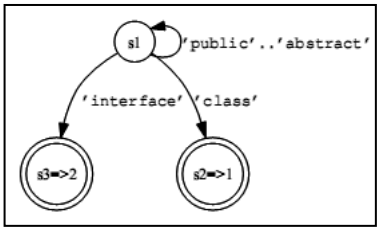
$$Sem\ \frac{\pi_i(\mathbb{S}) \quad A \rightarrow \{\pi_i\}?\ \alpha_i}{(\mathbb{S}, uA\delta) \overset{\pi_i}{\Longrightarrow} (\mathbb{S}, u\alpha_i\delta)} \qquad Syn\ \frac{(\mathbb{S}, A'_i) \Rightarrow^* (\mathbb{S}', w) \quad w \preceq w_r \quad A \rightarrow (A'_i)\text{=>}\alpha_i}{(\mathbb{S}, uA\delta) \overset{A'_i}{\Longrightarrow} (\mathbb{S}, u\alpha_i\delta)}$$

$$Closure\ \frac{(\mathbb{S}, \alpha) \overset{\lambda}{\Longrightarrow} (\mathbb{S}, \alpha'),\ (\mathbb{S}, \alpha') \overset{\vec{\lambda}}{\Longrightarrow}{}^*(\mathbb{S}, \beta)}{(\mathbb{S}, \alpha) \overset{\lambda\vec{\lambda}}{\Longrightarrow}{}^*(\mathbb{S}, \beta)}$$

Gambar 2. Predikat grammar dengan aturan turunan terkiri (Leftmost derivation rule)

LL(*) tidak merubah strategi *recursive descent parser*, dia hanya memperluas kemampuan memprediksi keputusan atas predikat pada LL. LL(*) dapat di pakai untuk membangun grammar dan secara otomatis melakukan *left-factoring* untuk mengenerate keputusan yang efisien. LL(*) secara otomatis akan menentukan kedalaman *lookahead*, di bandingkan dengan LL(*k*), dengan *k* lookahead. LL(*) juga mengijinkan *cyclic* DFA seperti pada gambar 3, DFA dengan loop yang dapat memeriksa urutan input *lookahead* yang membedakan alternatif. Perhatikan contoh rule dengan cyclic DFA di bawah ini

```
def : modifier* classDef
    | modifier* interfaceDef
    ;
```



Gambar 3. Cyclic DFA

Cyclic DFA dapat dengan mudah menghindari **modifier** menuju **class** atau **interface**, karena LL(*) akan menghasilkan skipping seperti gambar 4 berikut,

```
void def() {
  int alt=0;
  while (LA(1) in modifier) consume(); // scan past modifiers
  if ( LA(1)==CLASS ) alt=1;           // 'class'?
  else if ( LA(1)==INTERFACE ) alt=2;  // 'interface'?
  switch (alt) {
    case 1 : ...
    case 2 : ...
    default : error;
  }
}
```

Gambar 4. Teknik cyclic DFA pada LL(*)

Dalam kasus ini LL(*) akan melakukan loop sederhana yang di implementasikan pada rule **def** untuk memprediksi DFA. LL(*), juga mampu mengkonstruksi syntatic dan semantic predicate. Semantik mengacu pada sesuatu di belakang syntax atau semua hubungan antar simbol input kepada intepretasi statement. Perhatikan contoh rule dengan elemen matching lebih dari empat kali,

```
data :      BYTE BYTE BYTE BYTE
```

```
|       BYTE BYTE BYTE
|       BYTE BYTE
|       BYTE
;
```

Dengan CFG kita akan sukar menghitung kemungkinan kombinasinya (tanpa sematik predikat), saat kombinasinya melebihi empat, solusinya adalah dengan membuat semantik predikat

```
data: (b+=BYTE)+
{if($b.size()>4)«error»;};
atau
data: ( b+=BYTE )+ {$b.size()<=4}?;
```

Syntaic predicate sangat berguna pada dua situasi, saat LL(*) tidak dapat menangani grammar dan saat terdapat pecedence diantara dua alternatif yang ambigu.
Di lain pihak grammar merupakan scanner token dengan teknologi LL(*) untuk menangani dan mengenali valid input. Translasi suatu bahasa ke bahasa lain, membutuhan teknik yang di pakai di sini yaitu string template. String Template (ST) [12,14] merupakan engine template dan file template yang di pakai bersama-sama sebagai *controller* untuk melakukan translasi. ST merupakan DSL untuk mengenerate teks terstruktur dari internal struktur data untuk output suatu grammar. ST program dapat di tulis dalam java yang merupakan controller dalam finite state automata. Struktur ST dapat terdiri sebagai berikut pada gambar 5,

```
group groupname;
template1(a1, a2, ..., an) ::= "..."
...
```

Gambar 5. Struktur Group Template

Template berisi kumpulan referensial mutual pada output yang menyediakan pustaka untuk mengkonstruksi output bagi kontroler. Template di kompilasi menjadi instance bertipe string template yang bertindak sebagai prototype instance selanjutnya. Setiap instance template yang berisi suatu tabel pada instance tertentu dengan pasangan nilai dan nama atribut dan sebuah struktur abstract syntax tree (AST) yang di share oleh seluruh instance yang mewakili literal dan ekspresi. AST yang tersedia di pakai untuk mempercepat interpretasi dari template selama program berjalan. Himpunan atribut dalam tabel atribut di batas pada daftar argumen formal a_i. Template merupakan fungsi yang memetakan suatu atribut atau kumpulan atribut ke atribut lainnya dan menspesifikasi melalui pemilihan daftar literal output, t_i, dan ekspresi e_i, dengan demikian fungsi a_i seperti gambar 6 berikut,

$$F(a_1, a_2, ..., a_m) ::= \text{“}t_0 e_0 ... t_i e_i t_{i+1} ... t_{n_t} e_{n_e}\text{”}$$

Gambar 6. Fungsi String Template

Dimana t_i, mungkin string kosong, e_i terbatas pada sintak dan komputasional untuk memperkuat pembagian model-view. Notasi e_i berada dalam tanda kurung **<..>** yang megelilinginya. Jika tidak terdapat e_i maka template hanya terdiri literal tunggal saja, yaitu t_i, . Operator konkatenasi yang tidak terlihat di aplikasikan pada tiap elemen selanjutnya, evaluasinya di picu dengan konverter method `toString()`[14].

2. Pembahasan

Translator notasi algoritmik di implementasikan dengan ANTLR dan string template dalam bentuk suatu editor dan command line, yang di namakan Editor Translator Notasi Algoritmik (ETNA). Model notasi algoritmik yang di pilih sudah di ajarkan di lingkungan universitas, arsitektur yang sudah di kembangkan juga telah di tentukan [8,15]. Pendekatan yang di pakai dalam implementasi arsitektur ini adalah MVC (Model View Controller) berbasis pada paradigma object oriented yang di tulis dalam java sebagai target aplikasi. Grammar *Algoritmik.g* ditulis sebagai pengenal input yang akan di generate sebagai parser dan lexer dengan teknik LL(*), begitu juga di tulis kode group template *Algoritmik*.stg yang bertindak sebagai kontroler, yang mentranslasikan input token ke bahasa yang dikehendaki (bahasa c), berikut potongan grammar yang sudah di tulis pada gambar 7.

```
grammar Algoritmik;
......
Program
...
: declaration+
  -> program(
     libs={$program::libs},
     globals={$program::global
     functions={$program::functions},
     mainfunctions={$program::mainfunctions}
);
....
COMMENT
:'/*'(options{greedy=false;}:.)*'*/'
{$channel=HIDDEN;}
;
LINE_COMMENT
        :'//'~('\n'|'\r')*'\r'?'\n'
             {$channel=HIDDEN;}
```

Gambar 7. Grammar Algoritmik.g

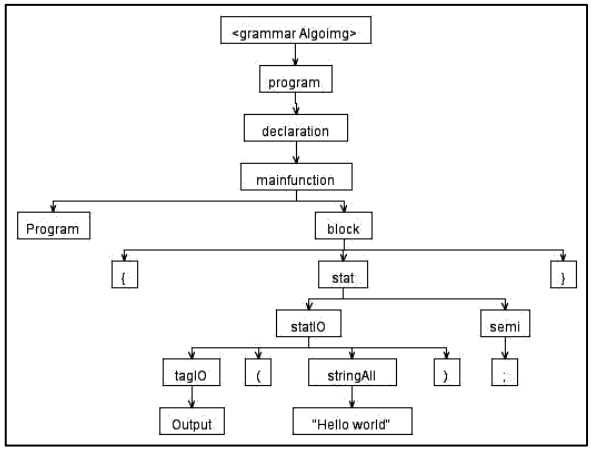
Group template dari grammar di atas di sajikan dalam potongan kode pada gambar 8 berikut

```
group Algoritmik;
program
(libs,globals,functions,mainfunctions)
::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
                    >>
...
```

Gambar 8. Template Algoritmik.stg

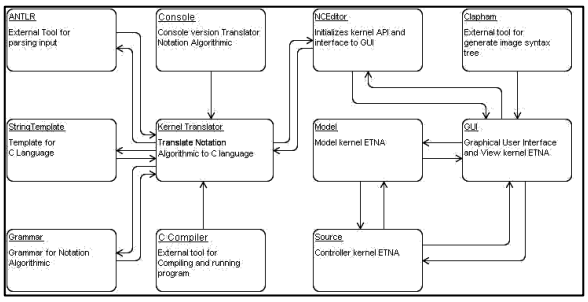
Output abstract syntax tree yang di hasilkan seperti pada gambar 9 berikut, misalkan di berikan input seperti berikut,

```
Program {
Output ("Hello World");
}
```



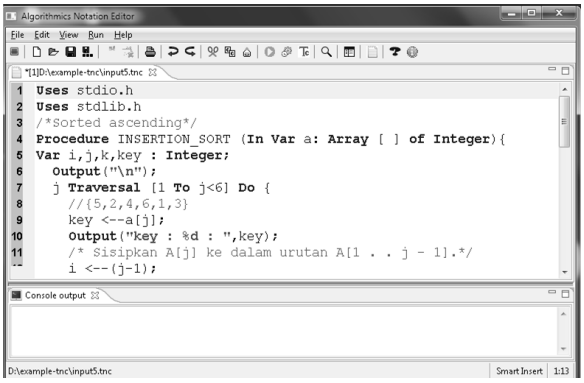
Gambar 9. Abstract Syntax Tree dari Grammar

Pembangunan editor berdasarkan pendekatan MVC dalam java seperti diagram block pada gambar 10 berikut,



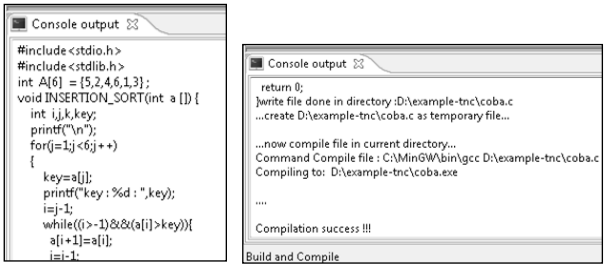
Gambar 10. Block Diagram ETNA

Block Diagram disini di gunakan untuk menjelaskan detail ETNA, yang terdiri dari kumpulan paket dan kelas yang terintegrasi Seperti terlihat, Translator (kernel ETNA), saling berkomunikasi dengan ANTLR, String Template dan grammar, sebagai paket dan kelas yang di pakai kernel. Parser dan Lexer dari grammar yang di hasilkan ANTLR, serta string template yang di tulis khusus untuk bahasa c (disesuaikan kebutuhan), dipakai oleh kernel selama ETNA berjalan. GUI sebagai interface ETNA dan user memakai kernel saat diperlukan. Console merupakan translator dalam versi command line yang memakai kernel serta kompiler c sebagai tool luar untuk menghasilkan file eksekusi juga di pakai oleh kernel. Interaksi kernel dan GUI (ETNA), melalui NCEditor, saat aplikasi dimulai kernel akan di inisialisasikan oleh NCEditor bersama-sama GUI sekaligus sebagai viewer, model dokumen serta source controller, sebagai implementasi model MVC. Sementara tool dari luar Clapham di pakai menggenerate image syntax tree yang saat ini di pakai untuk membantu user memahami notasi agoritmik (ke depan akan di manfaatkan untuk error trace secara visual). Seperti terlihat Kernel dan GUI tidak berkomunikasi secara langsung, tapi melalui Model dan Source yang di hubungkan oleh NCEditor untuk berkomunikasi dengan Kernel, dimana parser, lexer serta string template juga melalui kernel dan NCEditor untuk berkomunikasi dengan GUI sebagai interface user. Setelah di implementasikan dengan java ETNA berhasil di kembangkan dan sukses melewati beberapa uji fungsional, berikut tampilan ETNA pada gambar 10.

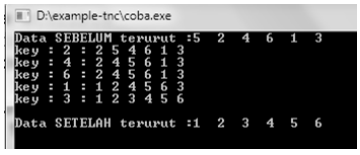


Gambar 10. Tampilan ETNA dengan file notasi algoritmik aktif

Sementara hasil translasi oleh ETNA dapat di lihat pada gambar 11 a, b dan c berikut ini,



(a) (b)



(c)

Gambar 11 (a). Hasil translasi, (b) Hasil kompilasi, (c). Hasil eksekusi

Masing-masing gambar (a) merupakan hasil translasi dari notasi algoritmik, sedang (b) saat ETNA melakukan kompilasi file hasil translasi menjadi executable file dan (c) saat ETNA menjalankan file hasil kompilasi dan sukses.

3. Kesimpulan

Dari hasil implementasi translator notasi algoritmik dengan teknik parsing LL(*) dan string template dalam

suatu aplikasi sistem translator notasi algoritmik, penulis dapat menyimpulkan sementara bahwa dari model translator yang di pilih dapat membantu pemakai (mahasiswa) untuk memecahkan masalah pemrograman dasar dalam bentuk notasi, tanpa perlu memahami bahasa yang di pakai. pembelajaran pemrograman dasar. *ETNA* berhasil dibangun dan sudah memiliki fitur yang cukup untuk membantu dalam menulis notasi (*code completion*, *syntax highlight*, *error correction*). Kemampuan mentranslasikan, mengkompilasi dan dapat menjalankan program *on the fly* juga lebih tepat dapat memberi informasi mengenai kegagalan atau keberhasilan solusi yang di tulis dengan notasi algoritmik. Kedepan *ETNA* ini perlu di lengkapi representasi *visual syntax tree* mengenai notasi algoritmik, yang muncul ketika di temukan kesalahan kode dan memunculkan grafik notasi syntax tree tepat di mana letak kesalahan di temukan serta bagaimana notasi yang benar, sehingga pengguna dapat memahami *syntax* yang benar dan segera membetulkan kesalahannya

Daftar Pustaka

[1] Blass, Andreas; Gurevich, Yuri., 2003, Algorithms: A Quest for Absolute Definitions, Bulletin of European Association for Theoretical Computer Science.

[2] Chairmain Cilliers, Andre Calitz, Jean Greyling, 2005, *The Application of The Cognitive Dimension Framework for Notations as an Instrument for the Usability analysis of an Introductory Programming tool*, Alternation Journal, 12.1b, p 543-576 ISSN 1023-1757.

[3] Chen Shyi-Ming, Lin Chung-Hui, Chen Shi-Jay, 2005, *Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques*, International Journal of Applied Science and Engineering. 3, 2: 89-100.

[4] David Harel, Yishai A. Feldman, 2004 , *Algorithmics: the spirit of computing, Edition 3*, Pearson Education, ISBN 0-321784-0.

[5] Hindayati Mustafidah, 2007, *Prestasi Belajar Mahasiswa dalam Mata Kuliah Pemrograman Dasar Melalui Pembelajaran Kooperatif Model Jigsaw*, Paedagogia, Agustus jilid 10 No 2, hal. 126 – 131.

[6] Ian Somerville, 2011, *Software engineering, 9th edition*, Pearson Education, Addison-Wesly, Boston, Massachusetts.

[7] Kruskal J. B, Jr., 1956, *On the shortest spanning subtree of a graph and the traveling salesman problem*. Proceedings of the American Mathematical Society, 7, pp. 48-50.

[8] Liem, Inggriani, 2007, *Draft Diktat Dasar Pemrograman (Bagian Prosedural)*, ITB , Bandung, unpublished.

[9] Reenskaug, Trygve M.H., 1979, *MODELS - VIEWS - CONTROLLERS.* , XEROX PARC.

[10] Reenskaug, Trygve M.H., 1979, *THING-MODEL-VIEW-EDITOR an Example from a planningsystem.* , Xerox PARC technical note May 1979.

[11] Stanchfield, Scott. *Applying MVC in VisualAge for Java. JavaDude*. [Online] 1996 - 2009. diakses: 10-10-2012. http://javadude.com/articles/vaddmvc2/mvc2.html.

[12] Parr, Terrence, 2006, A Functional Language For Generating Structured Text, di akses 10-10-2013, 2006, http://www. cs. usfca. edu/parrt/papers/ST. pdf

[13] Parr, Terrence, Fischer, Kathleen S, 2011, LL(*) : The Foundation of the ANTLR Parser Generator, PLDI '11, Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation, ACM New York, NY USA, ISBN: 978-1-4503-0663-8

[14] Parr, Terrance, Fischer, Kathleen S, 2004, Enforcing Strict Model-View Separation in Template Engines, New York, New York, USA. ACM 1-58113-844-X/04/0005

[15] Wijanarto, Achmad Wahid Kurniawan, 2012, *Model Translator Algoritmik ke Bahasa C*, Prosiding Kommit, Komputer dan Sistem Intelijen, Vol 7, 464-472 ISSN 2302-3740.

[16] Yuwono Indro Hatmojo, Sigit Yatmono, 2009, *Peningkatan Prestasi Mata Kuliah Komputer Dasar Mahasiswa D3 Teknik Elektro FT UNY Menggunakan Metode Belajar Berbasis Masalah*, Jurnal edukasi@Elektro Vol. 5, No.1, Maret, hal. 67 – 78.

[17] Alfred V Aho, Monica S Lam, Ravi Sethi , Jeffrey D Ullman. 2007. *Compilers : principles, techniques, and tools Second Edition.* New York : Pearson Education Addison Wesly, 2007.

[18] Alverd V Aho, Jeffery D Ullman. 1973. *The Theory of Parsing, Translation and Compiling.* New York : Prentice Hall Englewood Cliffs, 1973. 0-13-914564-8 .

[19] Andrew W Appel, Maia Ginsburg. 1998. *Modern Compiler Implementation In C.* New York : CAMBRIDGE UNIVERSITY PRESS, 1998.

[20] David A Watt, Deryck F Brown. 200. *Programming Language Processors in Java, Compiler and Intepreter.* New York : Pearson Education, Addison Wesly, 200.

[21] Dimitriev, Sergey. 2004. Language Orintation Programming : The Next Programming Paradigm. [November] s.l. : JetBrain, 2004.

[22] Fowler, Martin. 1999. *Analysis Patterns : Reuseable Object Models.* New York : Addison Wesley, 1999.

[23] —. 2010. *Domain Specific Languages* . New York : Addison-Wesley Professional , 2010. ISBN-10: 0-321-71294-3 .

[24] Gijzel, Bas van. 2009. *Comparing Parser Construction Techniques.* s.l. : University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, 2009.

[25] Hanson, Ralph E. Griswold and David R. 1980. *An Alternative to the Use of Patterns in String Processing.* Vol. 2 Issue 2 Pages 153-172 : ACM Transactions on Programming Languages and Systems, 1980.

[26] John R. Levine, Tony Mason, Doug Brown. 1992. *lex & yacc* . California : O'Reilly & Associates, Inc. 103 Morris Street, Suite A Sebastopol, CA 95472 , 1992. ISBN: 1-56592-000-7 .

[27] Keith D Cooper, Linda Torczon. 2003. *Engineering a Compiler, Second Edition.* San Francisco : ISBN: 1-56592-000-7 , 2003. ISBN-13: 978-1558606982 .

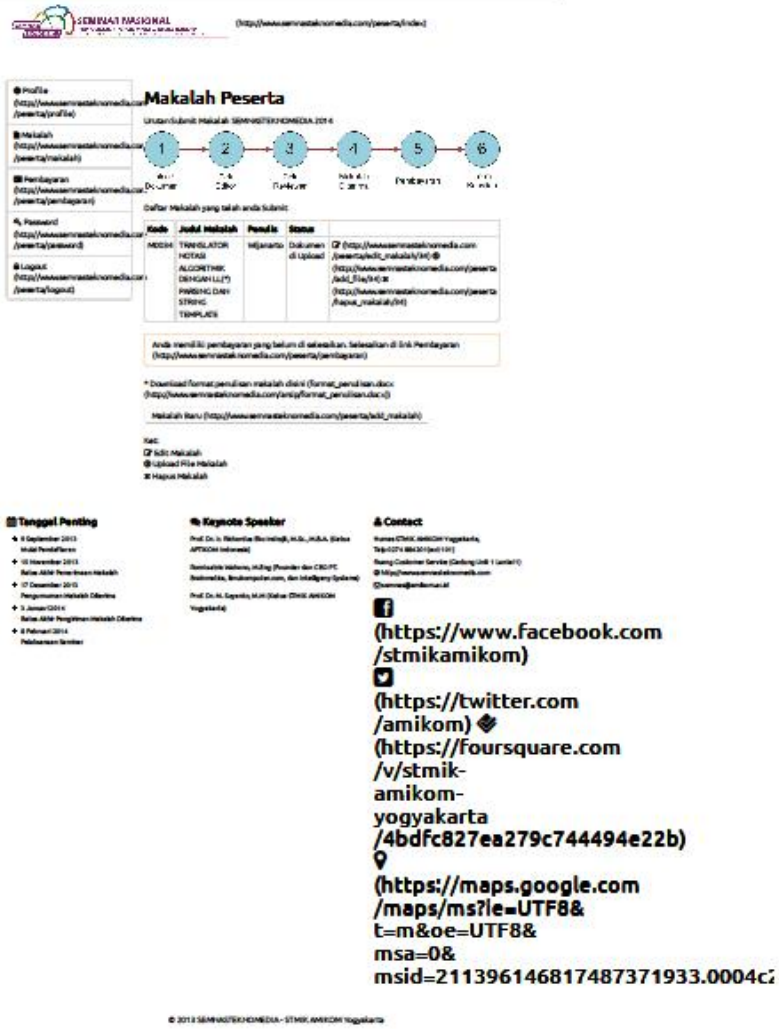
[28] P. Rechenberg, H. Mocchenbock. 1989. *A Compiler Generator For Microcomputers.* London : Prentice Hall International UK, 1989. ISBN : 0-13-155060-8.

Biodata Penulis

Wijanarto, Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Ilmu Komputer Universitas Gajah Mada Yogyakarta, lulus tahun 2006. Saat ini menjadi Dosen di UDINUS Semarang.

Ajib Susanto, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika UDINUS Semarang, lulus tahun 2004. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika Universitas Dian Nuswantoro Semarang, lulus tahun 2008.Saat ini menjadi Dosen di UDINUS Semarang.

Lampiran 3 Bukti Pengiriman Makalah



Makalah Peserta

1 2 3 4 5 6

Kode	Judul Makalah	Penulis	Status	
MDDM	TRANSLATOR NOTASI ALGORITMIK DENGAN LL(*) PARSING DAN STRING TEMPLATE	Hijianto	Dokumen di Upload	

Tanggal Penting

Keynote Speaker

Contact

(https://www.facebook.com/stmikamikom)
(https://twitter.com/amikom)
(https://foursquare.com/v/stmik-amikom-yogyakarta/4bdfc827ea279c744494e22b)
(https://maps.google.com/maps/ms?ie=UTF8&t=m&oe=UTF8&msa=0&msid=211396146817487371933.0004c

Lampiran 4 Jurnal Techno Com

Aspek Pedagogik Implementasi Translator Notasi Algoritmik Berbasis Parsing LL(*) dan String Template

Wijanarto[1], Ajib Susanto[2]
[1,2]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Nakula 5 - 11, Semarang, 50131, 024-3517261
E-mail : wijanarto.udinus@gmail.com[1], ajibsusanto@gmail.com[2]

Abstrak

Pengajaran pemrograman dasar pada tahun pertama, merupakan matakuliah dasar wajib bagi mahasiswa ilmu komputer. Algoritma merupakan model untuk memecahkan masalah di bidang pemrograman yang di implementasikan dalam bahasa pemrograman. Tidak mudah bagi seseorang dalam membuat solusi dalam bentuk bahasa formal, selain pemilihan alat atau aplikasi yang tepat untuk membantunya. Paper ini hasil penelitian yang melihat aspek pedagogik dari pengajaran pemrograman dengan model notasi algoritmik yang akan menghasilkan Domain Specific Language (DSL) untuk pengajaran pemrograman dasar. Implementasi model pengajaran di wujudkan dengan metode parsing LL(*) dan string template, yang otomatis akan mentranslasikan notasi algoritmik menjadi bahasa c standar. Model notasi algoritmik yang di pilih sudah pernah diterapkan dan diajarkan di perguruan tinggi. Grammar dihasilkan dengan bantuan ANTLR dan string template, yang di sesuaikan dengan model yang di pilih. Metode eksperimen di pakai untuk mengukur apakah terdapat perbedaan mahasiswa yang menggunakan translator dengan yang tidak menggunakan translator. Hasil dari penelitian ini berupa translator yang dapat di jalankan dalam command prompt serta editor gui yang di sebut ETNA (Editor Translator Notasi Algoritmik) serta rekomendasi penggunaan alat yang tepat dalam pengajaran pemrograman dasar. Alat ini diharapkan membantu seseorang atau mahasiswa mendisain solusi dalam bentuk notasi algoritmik, tanpa memikirkan kerumitan dalam bahasa yang pakai.

Kata Kunci: *parsing LL(*), aspek pedagogik, translator,algoritmik, experimental method*

Abstract

Teaching basic programming in the first year , a compulsory basic course for computer science students. Algorithm is a model for solving problems in the field of programming is implemented in a programming language . Not easy for someone to create a solution in the formal language form , in addition to the selection of the right tool or application for help. This paper studies looking at the results of the pedagogical aspects of teaching programming model algorithmic notation which will result in Domain Specific Language (DSL) for teaching basic programming. Implementation of the teaching model embodied by the method of parsing LL (*) and string template, which automatically translates an algorithmic notation became a standard C language. Notation algorithmic models have been implemented in select and taught in college . Generated with the help of ANTLR grammar and string templates, which are customized to the selected models . Experimental methods in use to gauge whether there are differences in the use of students who did not use a translator to translator . Results from this research is a translator that can be run in a command prompt and gui editor that called ETNA (Algorithmic Notation Editor Translator) as well as recommendations appropriate use of tools in the teaching of basic programming . This tool is expected to help a person or a student design algorithmic solutions in the form of notation , without thinking about the complexity of the language used .

.***Keywords:*** *LL (*), aspects pedagogy, translator, Algorithmics, experimental method*

1. PENDAHULUAN

Studi yang pernah dilakukan di Afrika Selatan menunjukan bahwa keberhasilan suatu pembelajaran pemrograman dasar di pengaruhi oleh 3 aspek, *pertama*, lingkungan belajar (alat atau aplikasi) yang mendukung notasi yang sederhana, yang dapat mengkonstruksi notasi umum untuk bahasa pemrograman.

K*edua*, penampilan visual dari struktur program harus memungkinkan mahasiswa pemrograman dasar dapat memahami semantik konstruksi program dan *ketiga*, lingkungan kerja aplikasi harus melindungi mahasiswa untuk tidak melakukan interpretasi dan pemahaman yang salah. Sesederhana apapun, suatu masalah pemrograman yang harus di pecahkan tetap dilakukan secara terstruktur dan ilmiah. Di bidang ilmu komputer atau teknik informatika, langkah atau urutan langkah pemecahan masalah atau metode yang logis, terstruktur dan berhingga di sebut sebagai algoritma [1,4]. Algoritma merupakan metode penyelesaian masalah yang umum dan banyak di lakukan hampir di seluruh bidang ilmu[3].

Di lain pihak pemahaman mahasiswa atau orang yang tertarik mempelajari pemrograman sering terkendala oleh bagaimana menggunakan bahasa itu sendiri. Di Indonesia studi mengenai pembelajaran pemrograman dasar sangat sedikit, apalagi yang menyangkut alat penunjang atau ketepatan penggunaan aplikasinya. Dalam penelitian yang di lakukan Hidayanti [5], lebih menyoroti metode pembelajaran dari aspek pedagogik, di mana capaian mahasiswa dalam belajar pemrograman dasar sangat rendah di karenakan rendahnya partisipasi, keaktifan dalam berdiskusi dan bertanya serta menjawab pertanyaan dalam kuliah. Sedangkan peneliti lain [13], dalam matakuliah sejenis yaitu komputer dasar, menyimpulkan (masih dari aspek pedagogik) bahwa metode belajar berbasis pada masalah dapat meningkatkan pemahaman materi dan prestasi mahasiswa, namun hanya efektif di lakukan dalam satu siklus saja.

Dengan demikian menurut hemat kami, dalam rangka mempermudah proses pembelajaran siswa dalam pemrograman dasar diperlukan model yang dapat menyederhanakan struktur dan semantik instruksi [2], sehingga dapat mempermudah pemahanan serta mengurangi interpretasi yang salah dalam rangka menyelesaikan masalah dalam bidang pemrograman.

Model sederhana yang diimplementasikan merupakan suatu translator notasi algoritmik yang secara otomatis dapat menghasilkan suatu bahasa pemrograman tingkat tinggi yang umum [12]. Sementara notasi algoritmik yang standar yang diberikan merupakan notasi yang sudah di ajarkan di perguruan tinggi [8]. Paper ini akan mengimplementasikan model yang di pilih untuk menghasilkan suatu translator notasi algoritmik ke dalam bahasa C standard dan mengukur tingkat perbedaan yang terjadi dalam pengajaran pemrograman dasar dari aspek pedagogik.

2. METODE, MODEL DAN ARSITEKTUR NOTASI ALGORITMIK

2.1 Parsing LL(*)

Parsing LL(*) merupakan perbaikan dar LL(k) untuk k>1, lookahead pada LL(k) terbatas pada k saja, sedangkan dalam LL(*) dapat mengestimasikan berapa kedalaman lookahead. Gambar 1 berikut selengkapnya mengenai notasi predikat grammar pada LL (*).

$A \in N$	Nonterminal
$a \in T$	Terminal
$X \in (N \cup T)$	Grammar symbol
$\alpha, \beta, \delta \in X^*$	Sequence of grammar symbols
$u, x, y, w \in T^*$	Sequence of terminals
$w_r \in T^*$	Remaining input terminals
ϵ	Empty string
$\pi \in \Pi$	Predicate in host language
$\mu \in \mathcal{M}$	Action in host language
$\lambda \in (N \cup \Pi \cup \mathcal{M})$	Reduction label
$\vec{\lambda} = \lambda_1..\lambda_n$	Sequence of reduction labels
Production Rules:	
$A \rightarrow \alpha_i$	i^{th} context-free production of A
$A \rightarrow (A'_i) \Rightarrow \alpha_i$	i^{th} production predicated on syntax A'_i
$A \rightarrow \{\pi_i\}? \ \alpha_i$	i^{th} production predicated on semantics
$A \rightarrow \{\mu_i\}$	i^{th} production with mutator

Gambar 1. Notasi Predikat Grammar LL(*)

Definisi formal dari LL(*) [14], sebagai berikut grammar *G=(N,T,P,S,Π,M)*, dimana *N* adalah himpunan non terminal simbol atau rule, T adalah himpunan terminal simbol atau token, *P* adalah himpunan produksi, *S∈N* merupakan start simbol, *Π* himpunan side effect free predikat semantik, dan *M* adalah himpunan aksi (mutator).

2.2 String Template

String Template (ST) [15,16] merupakan engine template dan file template yang di pakai bersama-sama sebagai controller untuk melakukan translasi. ST merupakan DSL untuk mengenerate teks terstruktur dari internal data struktur untuk output suatu grammar. ST program dapat di tulis dalam java yang merupakan controller dalam finite state automata. Struktur ST dapat terdiri sebagai berikut pada gambar 5,

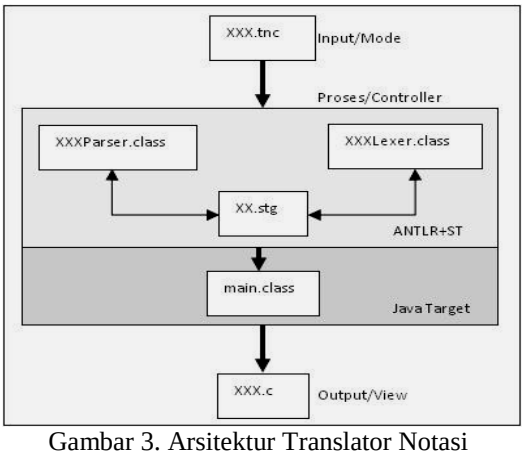
```
group groupname;
template1(a1, a2, ..., an) ::= "..."
...
```

Gambar 2. Struktur Group Template

Template berisi kumpulan referensial mutual pada output yang menyediakan pustaka untuk mengkonstruksi output bagi kontroler. Template di kompilasi menjadi instance bertipe string template yang bertindak sebagai prototype instance selanjutnya.

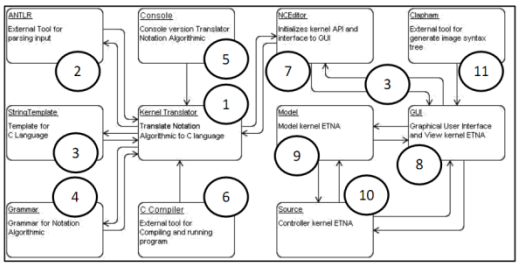
2.3 Arsitektur Aplikasi

Suatu sistem aplikasi di kembangkan dengan suatu metode atau cara yang beragam, paper ini akan menggunakan pendekatan Model View Contrller (MVC) [9,10,11]. Arsitektur Aplikasi adalah seperti gambar 3 sebagai berikut,



Gambar 3. Arsitektur Translator Notasi Algoritmik berbasis MVC

Input yang berupa file text dalam bentuk notasi standar algoritma akan di baca oleh scanner yang sesuai dengan grammar yang di generate oleh ANTLR. String Template merupakan translator (*hand coded*) notasi ke bahasa yang di spesifikasikan secara simultan saat membuat grammar. Generator notasi, yang menjadi test rig dalam bentuk class akan menghasilkan output bahasa yang valid. Sementara implementasi arsitektur dapat di lihat pada gambar 4 berikut,



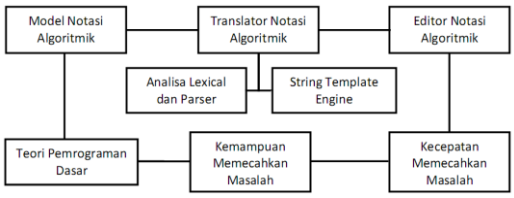
Gambar 4. Diagram Block Aplikasi

Translator (kernel ETNA) ditandai dengan lingkaran 1, saling berkomunikasi dengan ANTLR, lingkaran 2, String Template dan grammar di tandai lingkaran 3 dan 4, sebagai paket dan kelas yang di pakai kernel. Parser dan Lexer dari grammar yang di hasilkan ANTLR, serta string template yang di tulis khusus untuk bahasa c, dipakai oleh kernel selama ETNA berjalan. GUI sebagai interface ETNA dan user memakai kernel saat

diperlukan. Console, lingkaran 5 merupakan translator dalam versi command line yang memakai kernel serta kompiler c, lingkaran 6, sebagai tool luar untuk menghasilkan file eksekusi juga di pakai oleh kernel. Interaksi kernel dan GUI (ETNA) melalui NCEditor, ditandai lingkaran 7. Saat aplikasi dimulai kernel akan di inisialisasikan oleh NCEditor bersama-sama GUI sekaligus sebagai viewer ditandai lingkaran 8, model Translator (kernel ETNA) ditandai dengan lingkaran 9 serta source controller di tandai lingkaran 10, sebagai implementasi model MVC. Sementara tool dari luar Clapham lingkaran 11, menggenerate image syntax tree yang saat ini di pakai untuk membantu user memahami notasi agoritmik (ke depan akan di manfaatkan untuk error trace secara visual).

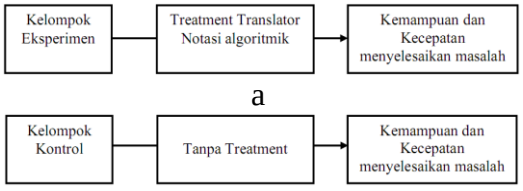
2.4. Kerangka Pikir Dan Paradigma Eksperimen [17]

Belajar pemrograman mahasiswa di harapkan tidak terjebak menggunakan bahasa pemrograman (program) yang cenderung rumit dan sukar di pahami, dan alat untuk membantu mahasiswa dalam belajar pemrograman adalah suatu bahasa natural (Notasi Algoritmik) yang mudah di mengerti dan mudah di terjemahkan oleh komputer dalam rangka menyelesaikan masalah di bidang pemrograman. Translator notasi algoritmik dapat membantu mahasiswa memecahkan masalah di bidang pemrograman dasar tanpa belajar bahasa program yang rumit, seperti kerangka pikir yang di jelaskan pada gambar 5 di bawah ini.



Gambar 5. Kerangka Pikir

Paradigma penelitian yang di pakai dalam metode eksperimen adalah seperti pada gambar 6 a dan b berikut,



a

b

Gambar 6. Paradigma Penelitian

Dari gambar paradigma penelitian di atas, variabel penelitian yang telah ditetapkan dikenal posttest dengan pengukuran penggunaan tanpa prates. Pembelajaran tanpa menggunakan translator notasi pada kelompok eksperimen dan pembelajaran dengan menggunakan translator notasi algoritmik untuk kelompok kontrol. Setelah itu, kedua kelompok tersebut dikenai pengukuran dengan menggunakan pascates. Sedang hipotesa yang di ujian adalah sebagai berikut :

a. Hipotesa Nol (Ho)

1) Tidak ada perbedaan penggunaan translator notasi algoritmik yang signifikan antara kelompok yang menyelesaikan masalah pemrograman dengan menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.

2) Penggunaan translator notasi algoritmik tidak lebih cepat memecahkan masalah pemrograman dibandingkan dengan tanpa menggunakan translator notasi.

b. Hipotesa Alternatif (Ha)

1) Terdapat perbedaan kemampuan menyelesaiakan masalah pemrograman antara kelompok yang menggunakan translator notasi algoritmik dan kelompok yang tanpa menggunakan translator notasi algoritmik.

2) Penggunaan translator notasi algoritmik lebih cepat memecahkan masalah pemrograman dibandingkan tanpa menggunakan translator notasi algoritmik dalam menyelesaikan masalah pemrograman dasar.

3. HASIL PENELITIAN

3.1 ETNA (Editor Translator Notasi Algoritmik)

ETNA atau editor translator notasi algoritmik merupakan hasil implementasi denga menggunakan metode parsing LL(*) dan string template dalam bentuk GUI. Potongan grammar yang di hasilkan dengan metode parsing LL(*) dan string template di sajikan dalam gambar 6 dan 7 berikut.

```
grammar Algoritmik;
......
Program
...
: declaration+
-> program(
    libs={$program::libs},
    globals={$program::global),
  functions={$program::functions},
mainfunctions={$program::mainfunctio
ns}
);
....
LINE_COMMENT
:'//'~('\n'|'\r')*'\r'?'\n'
{$channel=HIDDEN;}
;
```

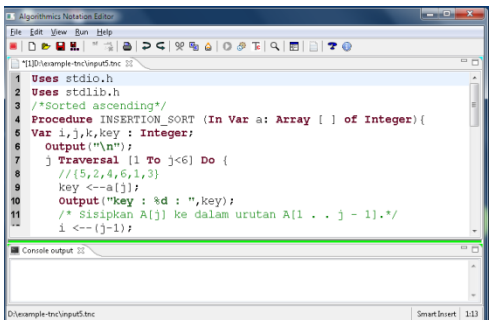
Gambar 6. Grammar notasi algoritmik

Sementara potongan string tempplate yang di hasilkan adalah sebagai berikut.

```
group Algoritmik;
program
(libs,globals,functions,mainfunctions
)
::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
...
```

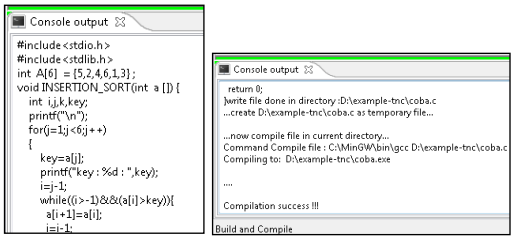
Gambar 7. String Template Algoritmik

Berikut tampilan ETNA saat sedang mengolah file source notasi algoritmik pada gambar 8.

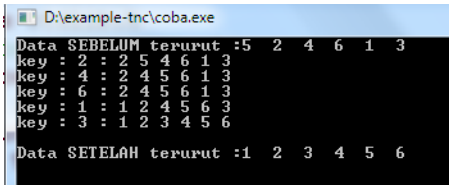


Gambar 8. ETNA

ETNA yang di kembangkan telah berhasil mentranslasikan notasi algoritmik, melakukan kompilai dan menjalankan file hasil kompilasi seperti terlihat pada gambar 9 berikut.



(a) (b)



(c)

Gambar 9 (a). Hasil translasi, (b) Hasil kompilasi, (c). Hasil eksekusi

3.2 Aspek Pedagogik

Hail uji normalitas sebaran data yang diperoleh dari pascates kemampuan dan kecepatan memecahkan masalah pemrograman dasar baik dari kelompok eksperimen maupun kontrol seperti pada tabel 1 dan 2 di bawah ini.

Tabel 1. Hasil Uji Normalitas Sebaran Data Pascates Kemampuan Menyelesaikan Masalah Pemrograman Dasar [Sumber : data primer]

Data	Sig.	Keterangan
Pascates Kelompok Kontrol	0.140	Sig. (2-tailed) > 0, 050 = normal
Pascates Kelompok Eksperimen	0.690	Sig. (2-tailed) > 0, 050 = normal

Kecepatan juga memperoleh hasil yang signifikan sebagai berikut,

Tabel 2. Hasil Uji Normalitas Sebaran Data Pascates Kecepatan Menyelesaikan Masalah Pemrograman Dasar [Sumber : data primer]

Data	Sig.	Keterangan
Pascates KK	0. 398	Sig. (2-tailed) > 0, 050 = normal
Pascates KE	0. 293	Sig. (2-tailed) > 0, 050 = normal

3.2.1 Perbandingan Data Kelompok Kontrol dan Kelompok Eksperimen

Berikut ini disajikan tabel perbandingan data pascatest skor tertinggi, skor terendah, *mean*, *median*, dan *mode* dari kelompok kontrol dan kelompok eksperimen. Tabel ini di buat untuk memberi kejelasan gambaran hasil penelitian yang di peroleh sedemikian rupa sehingga kita dapat memperbandingan dengan mudah.

Tabel 3. Perbandingan Data Statistik Pascates Kemampuan dan Kecepatan Menyelesaikan Masaah Pemrograman [Sumber : data primer].

	N	Min		Max		Mean		Median		Mode	
		A	B	A	B	A	B	A	B	A	B
X	38	29	60	85	21	64.68	44.58	64.00	46.33	61	56
Y	38	60	57	93	17	76.89	32.03	79.50	29.00	61	17

Keterangan :
A: Kemampuan ; B:Kecepatan
X : Pascatest Kelompok Kontrol
Y : Pascatest Kelompok Eksperimen

Sementara itu, hasil uji beda kelompok eksperimen dan control pada kemampuan dan kecepatan menyelesaikan masalah pemrograman dasar di sajikan dalam table 3 dan 4 berikut

Tabel 3. Hasil Uji-t Data Pascates Kemampuan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen [Sumber : data primer]

Data	T Hitung	df	P	Keterangan
A	4.638	74	.000	p < 0,05 = signifikan

Keterangan
A : Pascates kelompok kontrol dan kelompok eksperimen

Dari table di atas maka uji beda signifikan terdapat perbedaan kemampuan kelompok yang di uji. Pada hasil uji beda kecepatan menyelesaikan masalah pemrograman dasar juga member hasil yang signifian seperti di sajikan sebagai berikut

Tabel 4. Hasil Uji-t Data Skor Kecepatan Memecahkan Masalah Pemrograman Dasar Kelompok Kontrol dan Kelompok Eksperimen. [Sumber : data primer]

Data	T Hitung	df	P	Keterangan
A	-4.718	74	.000	p > 0,05 = signifikan

Keterangan
A : Pascates kelompok kontrol dan kelompok eksperimen

4. KESIMPULAN

Dari paparan di atas maka penulis menyimpulkan sementara bahwa implementasi model notasi algoritmik dengan LL(*) parsing dan string template berhasil di kembangkan menjadi suatu alat pembelajaran pemrograman dasar ETNA. Hasil uji beda terhadap ETNA dalam pemakaian pengajaran juga memberi hasil yang signifikan, sehingga dapat menerima hipotesa, bahwa terdapat perbedaan kemampuan dan kecepatan pemecahan masalah pemrograman dasar dengan ETNA pada kelompok eksperimen dan control. Kedepan perlu studi lebih lanjut mengenai pelacakan kesalahan pada ETNA dengan memanfaatkan kemampuan syntax tree, juga perlu di teliti lebih lanjut reaksi ETNA terhadap user dari aspek dimensi kognitif [2] maupun framework Nelson.

DAFTAR PUSTAKA

[29] Blass, Andreas; Gurevich, Yuri., 2003, *Algorithms: A Quest for Absolute Definitions*, Bulletin of European Association for Theoretical Computer Science.

[30] Chairmain Cilliers, Andre Calitz, Jean Greyling, 2005, *The Application of The Cognitive Dimension Framework for Notations as an Instrument for the Usability analysis of an Introductory Programming tool*, Alternation Journal, 12.1b, p 543-576 ISSN 1023-1757.

[31] Chen Shyi-Ming, Lin Chung-Hui, Chen Shi-Jay, 2005, *Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques*, International Journal of Applied Science and Engineering. 3, 2: 89-100.

[32] David Harel, Yishai A. Feldman, 2004 , *Algorithmics: the spirit of computing, Edition 3*, Pearson Education, ISBN 0-321784-0.

[33] Hindayati Mustafidah, 2007, *Prestasi Belajar Mahasiswa dalam Mata Kuliah Pemrograman Dasar*

Melalui Pembelajaran Kooperatif Model Jigsaw, Paedagogia, Agustus jilid 10 No 2, hal. 126 – 131.

[34] Ian Somerville, 2011, *Software engineering, 9th edition*, Pearson Education, Addison-Wesly, Boston, Massachusetts.

[35] Kruskal J. B, Jr., 1956, *On the shortest spanning subtree of a graph and the traveling salesman problem*. Proceedings of the American Mathematical Society, 7, pp. 48-50.

[36] Liem, Inggriani, 2007, *Draft Diktat Dasar Pemrograman (Bagian Prosedural)*, ITB , Bandung, unpublished.

[37] Reenskaug, Trygve M.H., 1979, *MODELS - VIEWS - CONTROLLERS.* , XEROX PARC.

[38] Reenskaug, Trygve M.H., 1979, *THING-MODEL-VIEW-EDITOR an Example from a planningsystem.* , Xerox PARC technical note May 1979.

[39] Stanchfield, Scott. *Applying MVC in VisualAge for Java. JavaDude*. [Online] 1996 - 2009. diakses: 10-10-2012. http://javadude.com/articles/vaddmvc2/mvc2.html.

[40] Wijanarto, Achmad Wahid Kurniawan, 2012, *Model Translator Algoritmik ke Bahasa C*, Prosiding Kommit, Komputer dan Sistem Intelijen, Vol 7, 464-472 ISSN 2302-3740.

[41] Yuwono Indro Hatmojo, Sigit Yatmono, 2009, *Peningkatan Prestasi Mata Kuliah Komputer Dasar Mahasiswa D3 Teknik Elektro FT UNY Menggunakan Metode Belajar Berbasis Masalah*, Jurnal edukasi@Elektro Vol. 5, No.1, Maret, hal. 67 – 78.

[42] Parr, Terrence, Fischer, Kathleen S, 2011, LL(*) : The Foundation of the ANTLR Parser Generator, PLDI '11, Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation, ACM New York, NY USA, ISBN: 978-1-4503-0663-8

[43] Parr, Terrence, 2006, A Functional Language For Generating Structured Text, di akses 10-10-2013, 2006, http://www. cs. usfca. edu/parrt/papers/ST. pdf

[44] Parr, Terrance, Fischer, Kathleen S, 2004, Enforcing Strict Model-View Separation in Template Engines, New York, New York, USA. ACM 1-58113-844-X/04/0005

[45] Sugiyono, 2010, Metode Penelitian Kuantitatif, Kualitatif dan R &F, Bandung, Alfabeta.

LAMPIRAN 5 Bukti Pengiriman Jurnal

FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

SURAT KETERANGAN
NO. 470/B-21/UDN-02/XI/2013

Yang bertanda tangan di bawah ini :

Nama : Wellia Shinta Sari, M.Kom

Bagian : Koordinator Suasana Akademik Fakultas Ilmu Komputer
Universitas Dian Nuswantoro

Menerangkan bahwa,

Nama : Wijanarto, M.Kom
Ajib Susanto, M.Kom

Judul Artikel : Aspek Pedagogik Implementasi Translator Notasi Algoritmik Berbasis Parsing LL(*) dan String Template

Telah melakukan submit artikel untuk Jurnal Techno Com.

Demikian surat keterangan ini kami buat, dan agar dapat di pergunakan sebagaimana mestinya.

Semarang, 10 Nopember 2013
Koordinator Suasana Akademik
Fakultas Ilmu Komputer

Wellia Shinta Sari, M.Kom
0686.11.1998.169

Kampus : Jl. Nakula I No. 5 - 11 Semarang 50131, Indonesia Telp. (024) 3569196 Fax. (024) 3569684 Ext.108
www.dinus.ac.id E-mail:sekretariat@fik.dinus.ac.id

Lampiran 6 File Grammar

```
grammar Algoritmik;
options {
      backtrack=true;
      memoize=true;
      k=2;
      language=Java;
      output=template;
}
scope slist {
    List locals;
    List stats;
    List funct;
}

@header {
      package org.wied.lang.grammar;
      import org.antlr.stringtemplate.*;
}

@lexer::header {
      package org.wied.lang.grammar;
}

program
scope {
    List libs;
      List globals;
      List functions;
      List mainfunctions;
      //List frealizations;
}

@init {
      $program::libs = new ArrayList();
      $program::globals = new ArrayList();
      $program::functions = new ArrayList();
      $program::mainfunctions = new ArrayList();
      //$program::frealizations = new ArrayList();
}
      :     declaration+
            -> program( libs={$program::libs},
                              globals={$program::globals},
                              functions={$program::functions},

      mainfunctions={$program::mainfunctions}//,

      //frealizations={$program::frealizations}
                              )
    ;
declaration
    : lib{$program::libs.add($lib.st);}
    | macro{$program::globals.add($macro.st);}
    | declType{$program::globals.add($declType.st);}
    | variable{$program::globals.add($variable.st);}
    | constant{$program::globals.add($constant.st);}
    | function{$program::functions.add($function.st);}
    | mainfunction{$program::mainfunctions.add($mainfunction.st);}
```

```
    //|
      frealization{$program::frealizations.add($frealization.st);}
    //|     COMMENT     ->{new StringTemplate($COMMENT.text);}
    //|     LINE_COMMENT->{new
StringTemplate($LINE_COMMENT.text);}
        ;
storage_class_specifier
      :     'Extern'    ->{new StringTemplate("extern");}
      |     'Static'    ->{new StringTemplate("static");}
      |     'Auto'            ->{new StringTemplate("auto");}
      |     'Register'  ->{new StringTemplate("register");}
      ;
primitiveType
      :     'Integer'   -> type_int()
    |   'Character' -> type_char()
    |   'Real'          -> type_float()
    | 'Double'    -> type_double()
      ;
extendedType
      :     'Short'           -> type_short()
      |     'Long'            -> type_long()
      |     'Signed'    -> type_signed()
      |     'Unsigned'  -> type_unsigned()
      ;
type //returns[String str]
    : 'String'    -> type_string()
      |     'Boolean'   -> type_bool()
    | structTag   -> {$structTag.st}
    | unionTag    -> {$unionTag.st}
    | enumTag     -> {$enumTag.st}
      |     ID                ->{new StringTemplate($ID.text);}
      |     extendedType* primitiveType   -
>extended(e={$extendedType.st},p={$primitiveType.st})
      //|   ident -> {$ident.st}
    ;
localType
      :     structType  -> {$structType.st}
      |     unionType   -> {$unionType.st}
      |     enumType    -> {$enumType.st}
      ;
/*
type
    : 'Integer'   -> type_int()
    |   'Character' -> type_char()
    |   'Real'          -> type_float()
    | 'String'    -> type_string()
    | 'Short'           -> type_short()
      |     'Long'            -> type_long()
      |     'Double'    -> type_double()
      |     'Signed'    -> type_signed()
      |     'Unsigned'  -> type_undigned()
      |     'Boolean'   -> type_bool()
    | structTag   -> {$structTag.st}
    | enumTag     -> {$enumTag.st}
    //|     advtypetag  -> {$advtypetag.st}
      |     declarator  -> {$declarator.st}
    ;

*/
```

```
typedeclarator
      : p+=ID (',' p+=ID)* -> getTypedeclarator(list={$p})
      ;
tagIO
      :     'Input'     ->inputstat()//{new
StringTemplate("scanf");}
      |     'Output'    ->outputstat()//{new
StringTemplate("printf");}
      ;
statIO
      :     tagIO '(' stringAll (','
actualParameters)?')'//paramE_or_PE )* ')'
            -
>statIO(tag={$tagIO.st},arg={$stringAll.st},exp={$actualParameters
.st})//{$paramE_or_PE.st})
      ;
stringAll
      :     STRING_LITERAL->{new
StringTemplate($STRING_LITERAL.text);}
      |     STRING_PASCAL->{new
StringTemplate($STRING_PASCAL.text);}
      ;
lib
      :     ('Uses' use+) ->setLib(lib={$use.st})
      ;
use
    :  //'"' ID '"'                        ->{new
StringTemplate("\""+$ID.text+"\"");}//->setFileID1(id1={$ID.text})
      STRING_LITERAL                ->{new
StringTemplate($STRING_LITERAL.text);}
    |  '<' ID '>'                   ->{new
StringTemplate("<"+$ID.text+">");}//->setFileID2(id2={$ID.text})
    |   PATH_LITERAL                ->{new
StringTemplate("\""+$PATH_LITERAL.text+"\"");}//-
>setFilesLit(plit={$PATH_LITERAL.text})
      |     '<' PATH_LITERAL '>'    ->{new
StringTemplate("<"+$PATH_LITERAL.text+">");}//-
>setFilesLib(plib={$PATH_LIBRARY.text})
      |    ID                       ->{new
StringTemplate("\""+$ID.text+"\"");}//->setFiles(id={$ID.text})
      ;
macro
      :     statMacro ->{$statMacro.st}
      ;
//#Def max(a,b)((a>b)? (a): (b))
statMacro //'Def' (type) ('As' (constSwitch)* -
>mdef(type={$type.text},val={$constSwitch.st})
      :     //'Def' ID ('As' replacement)? -
>mdef(id={$ID.text},r={$replacement.st})
            'Def' tagMacro ('As' replacement)?  -
>mdef(p={$tagMacro.st},r={$replacement.st})//mdef(id={$ID.text},r=
{$replacement.st})
      |     'IfNotDef' (replacement)                  -
>mifndef(r={$replacement.st})
      //|   'IfNotDef' (type) ->mifndef(type={$type.text})
      |     'EndDef' ->mendif()
      |     'ElseDef' ->melse()
      ;
tagMacro
```

```
      :     ( func_callStat | ID ) (primary_expression
(dot_or_rarrow+ | conditional_expression ))?
          //('('exprs')'(conditional_expression |
dot_or_rarrow+))?
                    -
>getPE(id={$ID.text},fc={$func_callStat.st},pe={$primary_expressio
n.st},

ce={$conditional_expression.st},dot={$dot_or_rarrow.st})
      //blom slese
      ;
replacement
      :     constSwitch             ->{$constSwitch.st}
      |     type                    ->{$type.st}
      ;

mainfunction
scope slist;
@init {
  $slist::locals = new ArrayList();
  $slist::stats = new ArrayList();
  $slist::funct = new ArrayList();
}
      :     'Program' block (function
{$slist::funct.add($function.st);})*
             ->
mainfunct(locals={$slist::locals},stats={$slist::stats},fr={$slist
::funct})
      ;
arg
      :     (argv','args) ->getArg(v={$argv.st},s={$args.st})
      ;
argv
      :     ('int' 'argv') ->{new StringTemplate("int argv");}
      ;
args
      :     ('char' '*' 'args' '[]') ->{new StringTemplate("char *
args [] ");}
      ;
/*
declType
      :     'Type' (options{backtrack=true;}:declarator? pointer?
'='? advtype? type?  ':'? structForm? unionForm? enumForm?
scalarForm? ident? )
                  -> {$function.size()>0 &&
$function::name==null}?

      globaldeclType(vars={$declarator.st},point={$pointer.st},aty
pe={$type.st},btype={$advtype.st},form={$structForm.st},ctype={$en
umForm.st},stype={$scalarForm.st},id={$ident.text})
                  ->
declType(vars={$declarator.st},point={$pointer.st},atype={$type.st
},btype={$advtype.st},form={$structForm.st},
ctype={$enumForm.st},stype={$scalarForm.st},id={$ident.text})
      ;
*/
declType
      :     'Type'
      (     declarator pointer? ':' advtype? type
```

```
      |     enumType
      |     structType
      |     unionType
      )     ->
declType(vars={$declarator.st},point={$pointer.st},atype={$type.st
},btype={$advtype.st},
            sform={$structType.st},
eform={$enumType.st},uform={$unionType.st})
      //|   localType -> {$localType.st}
      ;
getTagID
      :     (options{backtrack=true;}:ID? pointer*)   -
>getTagID(id={$ID.text},p={$pointer.st})
      ;
/*=====Tipe enum=====*/

enumType
      :     enumTag ':' enumForm ident?   -
>setEnumType(et={$enumTag.st},ef={$enumForm.st},id={$ident.text})
      ;

enumTag
      :     enumKey getTagID
            //->
enumTag(key={$enumKey.st},tag={$ID.text},point={$pointer.st})
            -> enumTag(key={$enumKey.st},tag={$getTagID.st})
      ;
enumKey
      :     'Enum' ->{new StringTemplate("enum");}
      ;
enumForm
      //:    '('val+=constSwitch(',' val+=constSwitch)*')'  -
>getenumVal(v={$val})
      :      '('val+=constSwitch(',' val+=constSwitch)*')'  -
>getenumVal(v={$val})
      ;
/*=====tipe struktur=====*/
structType
      :     structTag ':' structForm ident? -
>setStructType(st={$structTag.st},sf={$structForm.st},id={$ident.t
ext})
      ;
structTag
      :     structKey getTagID
            //->
structTag(key={$structKey.st},tag={$ID.text},point={$pointer.st})
            -> structTag(key={$structKey.st},tag={$getTagID.st})
      ;
structKey
      :     'Struktur'  ->{new StringTemplate("struct");}
      ;
structForm
      :     contents ->{$contents.st}
      ;
contents
      :     '<'f+=field(',' f+=field)*'>'->struktur(field={$f})
      ;
field
      :     variable ->getVar(field={$variable.st})
```

```
      ;
/*=====tipe union=====*/
unionType
      :     unionTag ':' unionForm ident? -
>setUnionType(ut={$unionTag.st},uf={$unionForm.st},id={$ident.text
})
      ;
unionTag
      :     unionKey getTagID
            -> unionTag(key={$unionKey.st},tag={$getTagID.st})
            //->
unionTag(key={$unionKey.st},tag={$ID.text},point={$pointer.st})
      ;
unionKey
      :     'Union'     ->{new StringTemplate("union");}
      ;
unionForm
      :     unioncontents ->{$unioncontents.st}
      ;
unioncontents
      :     '<'f+=field(',' f+=field)*'>'->union(field={$f})
      ;

//////////endoftype//////////////
tagName
      :     ID ->tagName(tag={$ID.text})
      ;
getTag
      :     listID* ->{$listID.st}
      ;
listID
      :     (options{backtrack=true;}:p+=ID (',' p+=ID)*)-
>getTag(tag={$p})
      ;


block
    :       '{' blockContents* '}'
    ;
blockContents
      :     localType {$slist::locals.add($localType.st);}
    | variable {$slist::locals.add($variable.st);}
    | constant {$slist::locals.add($constant.st);}
    | stat {$slist::stats.add($stat.st);}
    | LINE_COMMENT
    | COMMENT
    | WS
      ;
declarator
    :   //dd+=ID (',' dd+=ID)* ->setident(id={$dd})
    ident ->{new StringTemplate($ident.text);}
    ;

ident
      : dd+=ID (',' dd+=ID)* ->setident(id={$dd})
      ;
/*
type_qualifier
      :     'Constant' ->{new StringTemplate("const");}
```

```
      |     'Volatile' ->{new StringTemplate("volatile");}
      ;
*/
pointer
      :     p+=pointersymbol
(options{backtrack=false;k=1;memoize=true;}:p+=pointersymbol)* -
>pointer(a={$p})

      //(options{backtrack=false;memoize=false;}:p+=pointersymbol)
*
      ;
pointersymbol
      :     '^' -> pointersymbol()//{new StringTemplate("^")}
      ;
varOrStruct returns[String s]
      :     'Var' {$s="Var";}
      |     'Struktur' {$s="Struktur";}
      ;
isStruktur returns [boolean b]
    :       varOrStruct {if($varOrStruct.s.equals("Var")) $b=true;
else $b=false;}
    ;
initvar
    : (exprs  | '{' (p+=val_array (',' p+=val_array)*)? '}' )
    ;

val_array
      :     DECIMAL_LITERAL (',' DECIMAL_LITERAL)*
      |     FLOATING_POINT_LITERAL (',' FLOATING_POINT_LITERAL)*
      |     CHARACTER_LITERAL (',' CHARACTER_LITERAL)*
      |     STRING_LITERAL (',' STRING_LITERAL)*
      |     STRING_PASCAL (',' STRING_PASCAL)*
      ;
//blom di pake
advtypetag
      :     advtype type ->
      advtypetag(a={$advtype.st},b={$type.st})
      ;
advtype
      :     'Array' '[' ( p+=argArray (',' p+=argArray )* )* ']'
'of' -> getArrayForm(idx={$p})
      ;
argArray //getEndArray(endArray={$endArray.text})
      :     startArray ('To'|'..') endArray  ->
getEndArray(startArray={$startArray.text},endArray={$endArray.text
})
      ;

startArray
      :     exprs | DECIMAL_LITERAL
      ;
endArray
      :     exprs | DECIMAL_LITERAL
      ;

frealization
scope {
    String name;
}
```

```
scope slist;
@init {
  $slist::locals = new ArrayList();
  $slist::stats = new ArrayList();
}
    :   funcType+ ID {$function::name=$ID.text;}
        //'('  ( p+=formalParameter ( ',' p+=formalParameter )* )?
')' (semi|((':' type)* pointer*))
        // block*
        '('  ( p+=formalParameter ( ',' p+=formalParameter )* )?
')' ((':' type) pointer*)?
        block
        -> freal(funcType={$funcType.st},type={$type.st},
name={$function::name},

locals={$slist::locals},stats={$slist::stats},args={$p},
                    p={$pointer.st})
    ;
function
scope {
    String name;
}
scope slist;
@init {
  $slist::locals = new ArrayList();
  $slist::stats = new ArrayList();
}
    :   funcType+ ID {$function::name=$ID.text;}
        //'('  ( p+=formalParameter ( ',' p+=formalParameter )* )?
')' (semi|((':' type)* pointer*))
        // block*
        '('  ( p+=formalParameter ( ',' p+=formalParameter )* )?
')' ((':' type) pointer*)? semi?
        block?
        -> function(funcType={$funcType.st},type={$type.st},
name={$function::name},

locals={$slist::locals},stats={$slist::stats},args={$p},s={$semi.s
t},
                    p={$pointer.st})
    ;
funcType
      :     'Function'
      |     'Procedure' -> type_void()
      ;

inout
      :'Out' -> type_pointer()
      |'In'
      ;

formalParameter

    :   inout* 'Var' (v+=paramName (',' v+=paramName)*)? ':'
advtype* type
        ->
parameter(a={$advtype.st},type={$type.st},tinout={$inout.st},name=
{$v})
    ;
```

```
paramName
scope {
    String p;
}
      :     ID pointer* {$paramName::p=$pointer.text;}
            ->{$paramName::p==null}?
                  getParam(id={$ID.text})
            ->getParamPoint(id={$ID.text},point={$pointer.st})
      ;
funcName
      :
      (     ID                ->{new StringTemplate($ID.text);}
      |     alokasi           ->{$alokasi.st}
      |     retStat           ->{$retStat.st}
      )
      ;
retStat
      :     'Return'    ->{new StringTemplate("return");}
      ;
alokasi
      :     'MAlokasi'  ->{new StringTemplate("malloc");}
      |     'CAlokasi'  ->{new StringTemplate("calloc");}
      |     'MDealokasi' ->{new StringTemplate("free");}
      ;
func_callStat
      :     funcName'(' (actualParameters)? ')'
            ->functioncall(f={$funcName.st},
ap={$actualParameters.st})
      ;
actualParameters
      :     p+=paramE_or_PE (',' p+=paramE_or_PE)* -
>getAParam(p={$p})
      ;
paramE_or_PE
      :     exprs                         ->{$exprs.st}
      |     primary_expression      ->{$primary_expression.st}
      |     postfix_expression      ->{$postfix_expression.st}
      |     func_callStat           ->{$func_callStat.st}
      |     ID                            ->{new
StringTemplate($ID.text);}
      |     typecastexpr            ->{$typecastexpr.st}
      //|   NUMBER                        ->{new
StringTemplate($NUMBER.text);}
      ;
ctype
    : 'Integer'   -> type_int()
    |   'Character' -> type_char()
    |   'Real'          -> type_float()
    | 'String'    -> type_string()
    | 'Short'           -> type_short()
      |     'Long'            -> type_long()
      |     'Double'    -> type_double()
      |     'Signed'    -> type_signed()
      |     'Unsigned'  -> type_undigned()
      |     'Boolean'   -> type_bool()
    | structTag   -> {$structTag.st}
    | enumTag     -> {$enumTag.st}
    ;
//baru di pake di actualparameter
```

```
//di ekspresi blom
typecastexpr
      :     '(' ctype ')'
      (     ID
      |     func_callStat
      |     '('exprs')'
      )-
>typecastexpr(c={$ctype.st},id={$ID.text},f={$func_callStat.st},e=
{$exprs.st})
      ;

constant
scope {
    String atype;
}
      //storage_class_specifier
    :  'Constant'  declarator(pointer)* ':' (advtype)*  type  '='
initvar ';'
      {$constant::atype=$advtype.text;}
        -> {$function.size()>0 && $function::name==null}?

      globalConstant(pointer={$pointer.st},type={$type.st},name={$
declarator.st},idx={$advtype.st},init={$initvar.text})
        -> {$constant::atype==null}?

      initConst(pointer={$pointer.st},type={$type.st},name={$decla
rator.st},init={$initvar.text})
        -
>initConst(pointer={$pointer.st},type={$type.st},name={$declarator
.st},idx={$advtype.st},init={$initvar.text})
    ;
variable
scope {
    String initial;
    String atype;
}
      //storage_class_specifier
    :   varOrStruct isStruktur? declarator pointer* ':' advtype*
type*  ('<==' initvar )* ';'

      {$variable::initial=$initvar.text;$variable::atype=$advtype.
text;}
        -> {$isStruktur.b && $variable::initial==null &&
$variable::atype==null}?

      globalStruct(v={$varOrStruct.s},point={$pointer.st},name={$d
eclarator.st},idx={$advtype.st})
        -> {$function.size()>0 && $function::name==null}?

      globalVariable(point={$pointer.st},type={$type.st},name={$de
clarator.st},idx={$advtype.st},init={$initvar.text})
        ->{$variable::initial==null && $variable::atype!=null}?

      variable(point={$pointer.st},type={$type.st},name={$declarat
or.st},idx={$advtype.st})
       ->{$variable::initial==null && $variable::atype==null}?

      variable(point={$pointer.st},type={$type.st},name={$declarat
or.st})
```

```
        ->{$variable::initial!=null && $variable::atype==null}?

      initVar(point={$pointer.st},type={$type.st},name={$declarato
r.st},init={$initvar.text})
        -
>initVar(point={$pointer.st},type={$type.st},name={$declarator.st}
,idx={$advtype.st},init={$initvar.text})
    ;
stat
scope slist;
@init {
  $slist::locals = new ArrayList();
  $slist::stats = new ArrayList();
}
    :       forStat -> {$forStat.st}
    | ifstat  -> {$ifstat.st}
    | switchstat  ->{$switchstat.st}
    |       exprs semi -> statement(expr={$exprs.st})
    |       block -> statementList(locals={$slist::locals},
stats={$slist::stats})
    |       assignStat semi -> {$assignStat.st}
    | func_callStat semi ->procedurecall(f={$func_callStat.st})
    | statIO semi ->{$statIO.st}
    | break_or_cont semi ->setbreakcont(bc={$break_or_cont.st})
    |       semi -> {new StringTemplate(";");}
    ;
forStat
scope slist;
@init {
  $slist::locals = new ArrayList();
  $slist::stats = new ArrayList();
}
                        //d=exprs masalah utk assignSat
    :       ID* 'Traversal' ('[' s=logCond range e=logCond ']')*
('Step' exprs_or_assignStat)* 'Do'
              block ';'
      ->
forLoop(id={$ID.text},range={$range.b},start={$s.st},end={$e.st},s
tep={$exprs_or_assignStat.st},
                locals={$slist::locals}, stats={$slist::stats})
    |       'Repeat' block 'Until' '('(logCond)*')' ';'
      ->
repeatstat(e={$logCond.st},locals={$slist::locals},stats={$slist::
stats})
      |     'While' '('(logCond)*')' 'Do' block ';'
      ->
whilestat(e={$logCond.st},locals={$slist::locals},stats={$slist::s
tats})
      ;
exprs_or_assignStat
      :     logCond           ->{$logCond.st}
      |     lvalue assignOpr casting? rvalue ->
assignLoop(lval={$lvalue.st},asg={$assignOpr.st},c={$casting.st},e
={$rvalue.st})
      ;
            //|   lvalue assignOpr casting* rvalue ->
assignLoop(lval={$lvalue.st},asg={$assignOpr.st},c={$casting.st},e
={$rvalue.st})
```

```
opt returns[String s]
      :     'To' {$s="To";}
      |     'DownTo' {$s="DownTo";}
      ;
range returns [boolean b]
    :       opt {if($opt.s.equals("To")) $b=true; else $b=false;}
    ;
ifstat
      :     'If' '('(logCond)')' 'Then' thenstat ('Else'
elsestat)*
            -
>ifstat(log={$logCond.st},t={$thenstat.st},e2={$elsestat.st})
      ;
logCond
      :     exprs                           ->{$exprs.st}
      |     postfix_expression      ->{$postfix_expression.st}
      |     rvalue                          ->{$rvalue.st}
      |     constSwitch             ->{$constSwitch.st}
      |     primary_expression      ->{$primary_expression.st}
      |     ID                              ->{new
StringTemplate($ID.text);}
      ;
break_or_cont
      :     'Break'  -> {new StringTemplate("break");}
    | 'Continue'  -> {new StringTemplate("continue");}
      ;
thenstat
      :     stat ->getthenstat(s={$stat.st})
      ;

elsestat
      :     stat ->getelsestat(s={$stat.st})
      ;
switchstat
      :     'Depend On' '('ID')' '{'
                  (sw+=switchPart (';' sw+=switchPart)*)
                  elseswithcpart?
             '}' -
>switchstat(id={$ID.text},spart={$sw},elseswitch={$elseswithcpart.
st})
      ;
constSwitch
      :
      (     ID pointer* ('=' DECIMAL_LITERAL)?  ->
refVar(id={$ID.text},point={$pointer.st},n={$DECIMAL_LITERAL.text}
)
      |     constexpression                                 -
>{$constexpression.st}
      )
      ;
switchPart
      :     (constSwitch ':' stat) ->
getSwitchPart(id={$constSwitch.st},stat={$stat.st})
      ;
elseswithcpart
      :     ('Default' ':' stat) ->getElseswitch(stat={$stat.st})
//:   ('Default' constSwitch ':' stat) -
>getElseswitch(id={$constSwitch.st},stat={$stat.st})
      ;
```

```
assignStat
    : lvalue assignOpr  casting? rvalue ->
assign(lval={$lvalue.st},asg={$assignOpr.st},c={$casting.st},e={$r
value.st})
//    :     lvalue assignOpr  rvalue ->
assign(lval={$lvalue.st},asg={$assignOpr.st},e={$rvalue.st})
    ;

rvalue
      :     exprs ->{$exprs.st}
      |     postfix_expression      ->{$postfix_expression.st}
      |     unary_opr (exprs|postfix_expression)+ -
>unary_exprs(a={$unary_opr.st},e={$exprs.st},pe={$postfix_expressi
on.st})
      |     func_callStat ->{$func_callStat.st}
      |     DECIMAL_LITERAL                              -> {new
StringTemplate($DECIMAL_LITERAL.text);}
      ;
conditional_expression
      :     '?' '('b=rvalue')' ':' '('c=rvalue')' -
>condExprs(b={$b.st},c={$c.st})
      ;
lvalue
      :     unary_expression  ->{$unary_expression.st}
      |     postfix_expression      ->{$postfix_expression.st}
      ;
unary_expression
      :     postfix_expression                           -
>{$postfix_expression.st}
      |     'SizeOf' (type_or_unexp)            -
>sizeExprs(e={$type_or_unexp.st})
      ;
casting
      :     '('type pointer*')'     ->
castexprs(t={$type.st},p={$pointer.st})
      ;

type_or_unexp
      :     casting                         ->{$casting.st}
      |     '('unary_expression')'  ->{$unary_expression.st}
      ;
postfix_expression
      :     primary_expression
      (     '[' exprs ']'
      |     '[' ID ']'
      |     dot_or_rarrow+
      |     '(' argument_expression_list ')'
      |     conditional_expression
      |     func_callStat
      )*
      -
>postexprs(pe={$primary_expression.st},e={$exprs.st},id={$ID.text}
,dr={$dot_or_rarrow.st},

      al={$argument_expression_list.st},ce={$conditional_expressio
n.st},f={$func_callStat.st})
      ;
argument_expression_list
```

```
      :     aa+=assignStat (',' aa+=assignStat)* -
>argExprsList(al={$aa})
      ;
dot
      :     '.' postfix_expression  -
>setDot(id={$postfix_expression.st})
      ;
rarrow
      :     '->' postfix_expression -
>setArrow(id={$postfix_expression.st})
      ;
dot_or_rarrow
      :     dot         ->{$dot.st}
      |     rarrow      ->{$rarrow.st}
      ;
primary_expression
      :     ID pointer*                    ->
idp(id={$ID.text},p={$pointer.st})
      |     constexpression         -> {$constexpression.st}
      |     '('exprs')'                    ->
innerexpr(e={$exprs.st})//{$exprs.st}
      //{new StringTemplate("("+$exprs.text+")");}//{$exprs.st}
      ;

constexpression
      :     HEX_LITERAL                    -
>hexconstexpr(hex={$HEX_LITERAL.text})
      |     OCTAL_LITERAL                  -
>octconstexpr(oct={$OCTAL_LITERAL.text})
      |     DECIMAL_LITERAL                -
>decconstexpr(dec={$DECIMAL_LITERAL.text})
      |     CHARACTER_LITERAL       -
>chrconstexpr(chr={$CHARACTER_LITERAL.text})
      |     STRING_LITERAL                 -
>strconstexpr(str={$STRING_LITERAL.text})
      |     STRING_PASCAL                  -
>strconstexprpas(value={$STRING_PASCAL.text})
      |     FLOATING_POINT_LITERAL  -
>floconstexpr(flo={$FLOATING_POINT_LITERAL.text})
      //|   ID                             -> {new
StringTemplate($ID.text);}
      ;
assignFunction
      :     (ID|postfix_expression) pointer* assignOpr
func_callStat semi*
      ->
assignfunc(lhs1={$ID.text},pe={$postfix_expression.st},point={$poi
nter.st},asg1={$assignOpr.st},rhs1={$func_callStat.text},s={$semi.
st})
      ;
unary_opr
      :     '&'         -> {new StringTemplate("&");}
      |     '@'         -> {new StringTemplate("@");}
      |     '(+)' -> {new StringTemplate("+");}
      |     '(-)' -> {new StringTemplate("-");}
      |     'Negasi'-> oprunNeg()//{new StringTemplate("~");}
      |     'Not' -> oprunNot()//{new StringTemplate("!");}
      |     'Inc'       -> {new StringTemplate("++");}
      |     'Dec'       -> {new StringTemplate("--");}
```

```
      ;

assignOpr
      :     '<--' ->oprassignEq()// {new StringTemplate("=")}//
      |     '+=' -> {new StringTemplate("+=");}//opasumplus()
      |     '-=' -> {new StringTemplate("-=");}//opasummin()
      |     '*=' -> {new StringTemplate("*=");}//opasummul()
      |     '/=' -> {new StringTemplate("/=");}//opasumdiv()
      |     '%=' -> {new StringTemplate("\%=");}//opasummod()
      |     '<<=' -> {new StringTemplate("<<=");}//opasummod()
      |     '>>=' -> {new StringTemplate(">>=");}//opasummod()
      |     '&=' -> {new StringTemplate("&=");}//opasummod()
      |     '^=' -> {new StringTemplate("^=");}//opasummod()
      |     '|=' -> {new StringTemplate("|=");}//opasummod()
      ;
term
      :     ID                                    ->{new
StringTemplate($ID.text);}
      |     lvalue                                ->{$lvalue.st}
      |     postfix_expression            -
>{$postfix_expression.st}
      //|   exprs                                 -
>{$exprs.st}//innerexpr(e={$exprs.st})
//    |     CHARACTER_LITERAL              ->{new
StringTemplate($CHARACTER_LITERAL.text);}//cconst(value={$CHARACTE
R_LITERAL.text})
//    |     STRING_LITERAL                 ->{new
StringTemplate($STRING_LITERAL.text);}//sconst(value={$STRING_LITE
RAL.text})
//    |     STRING_PASCAL                  ->{new
StringTemplate($STRING_PASCAL.text);}//sconstpas(value={$STRING_PA
SCAL.text})
//    |     DECIMAL_LITERAL                ->{new
StringTemplate($DECIMAL_LITERAL.text);}//iconst(value={$DECIMAL_LI
TERAL.text})
//    |     FLOATING_POINT_LITERAL  ->{new
StringTemplate($FLOATING_POINT_LITERAL.text);}//fconst(value={$FLO
ATING_POINT_LITERAL.text})
      |     func_callStat                  ->{$func_callStat.st}
//    |     NUMBER                                -> {new
StringTemplate($NUMBER.text);}
    ;

postfix
      :     term (unary_opr)* ->
postfix(opr={$unary_opr.st},term={$term.st})
      ;

castid
      :     ('('ctype')')* postfix  ->
castid(c={$ctype.st},t={$postfix.st})
      ;

negation
      :     (unary_opr)* postfix -
>negasi(opr1={$unary_opr.st},term={$postfix.st})
      ;
unary
```

```
      :     (unary_opr)* negation -
>unary(opr1={$unary_opr.st},unary={$negation.st})
      ;
mult
      :     a=unary ((oprmult) b=unary)* -
>mult(op1={$a.st},opr={$oprmult.st},op2={$b.st})
      ;
oprmult
      :     '*'   ->{new StringTemplate("*");}//oprmultmul()
      |     'Div' ->oprmultdiv()//{new StringTemplate("/")}//
      |     'Mod' ->oprmultmod()//{new StringTemplate("\%")}//
      ;
add
      :     a=mult ((opradd) b=mult)* -
>add(op1={$a.st},opr={$opradd.st},op2={$b.st})
      ;
opradd
      :     '+' ->{new StringTemplate("+");}//opraddplus()
      |     '-' ->{new StringTemplate("-");}//opraddmin()
      ;
relation
      :     a=add ((oprrel) b=add)* -
>relation(op1={$a.st},opr={$oprrel.st},op2={$b.st})
      ;
oprrel
      :     '='   ->oprreleq()//{new StringTemplate("==")}//
      |     '<>'  ->oprrelneq()//{new StringTemplate("!=")}//
      |     '<'         ->{new StringTemplate("<");}//oprrellt()
      |     '<='  ->{new StringTemplate("<=");}//oprrelle()
      |     '>'         ->{new StringTemplate(">");}//oprrelgt()
      |     '>='  ->{new StringTemplate(">=");}//oprrelge()
      ;
shiftbit
      :     a=relation ((oprshift) b=relation)* -
>shiftbit(op1={$a.st},opr={$oprshift.st},op2={$b.st})
      ;
oprshift
      :     'shl' ->oprshl()//{new StringTemplate("<<")}
      |     'shr' ->oprshr()//{new StringTemplate(">>")}
      |     'shand'     ->oprsand()
      |     'shor'      ->oprsor()
      ;
exprs
      :     a=shiftbit  ((oprexpr) b=shiftbit )*  -
>exprs(op1={$a.st},opr={$oprexpr.st},op2={$b.st})
      ;
oprexpr
      :     'And'       ->oprexprsand()//{new
StringTemplate("&&")}//
      |     'Or'  ->oprexprsor()//{new StringTemplate("||")}//
      |     'Xor'   ->oprexprsxor()
      ;

semi
      :     ';'         ->{new StringTemplate(";");}
      ;
/*
NUMBER
      :     '0'..'9' ('0'..'9')*
```

```
      ;
*/
ID
      :     LETTER (LETTER|'0'..'9')*
      ;

PATH_LITERAL
    : ID('\\'|':'|ID)*
    ;

fragment LETTER
      :     '$'
      |     '.'
      |     'A'..'Z'
      |     'a'..'z'
      |     '_'
      ;
CHARACTER_LITERAL
      :     '\'' (EscapeSequence | ~('\'' | '\\')) '\''
      ;
STRING_PASCAL
      :     '\'' (EscapeSequence | ~('\\'|'\''))* '\''
      ;
STRING_LITERAL
      :     '"' (EscapeSequence | ~('\\'|'"'))* '"'
      ;
HEX_LITERAL
      :     '0' ('x'|'X') HexDigit+ IntegerTypeSuffix?
      ;
DECIMAL_LITERAL
      :     '0'..'9' ('0'..'9')*
      //('0'|'1'..'9' '0'..'9'*) IntegerTypeSuffix?
      ;
OCTAL_LITERAL
      :     '0' ('0'..'7')+ IntegerTypeSuffix?
      ;
fragment HexDigit
      :     ('0'..'9'|'a'..'f'|'A'..'F')
      ;
fragment IntegerTypeSuffix
      :     ('u'|'U')* ('l'|'L')
//    |     ('u'|'U')  ('l'|'L')*
      ;

FLOATING_POINT_LITERAL
      :     ('0'..'9')+ '.' ('0'..'9')* FloatTypeSuffix?
//Exponent? FloatTypeSuffix?
//    |     '.' ('0'..'9')+ Exponent? FloatTypeSuffix*
//    |     ('0'..'9')+ Exponent FloatTypeSuffix*
//    |     ('0'..'9')+ Exponent* FloatTypeSuffix
      ;
fragment Exponent
      :     ('e'..'E') ('+'|'-')? ('0'..'9')+
      ;
fragment FloatTypeSuffix
      :     ('f'|'F'|'d'|'D'|'l'|'L')
      ;
fragment EscapeSequence
      :     '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
```

```
      |     OctalEscape
      ;
fragment OctalEscape
      :     '\\' ('0'..'3')('0'..'7') ('0'..'7')
      |     '\\' ('0'..'7') ('0'..'7')
      |     '\\' ('0'..'7')
      ;
fragment UnicodeEscape
      :     '\\' 'u' HexDigit HexDigit HexDigit HexDigit
      ;
WS
      :     (' ' | '\r'|'\t'|'\u000C'|'\n') {$channel=HIDDEN;}
      ;
COMMENT
      :     '/*' (options {greedy=false;} : . )* '*/'
{$channel=HIDDEN;}
      ;
LINE_COMMENT
      :     '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
      ;
```

Lampiran 7 File Template

```
group Algoritmik;
program(libs,globals,functions,mainfunctions) ::=<<
<libs; separator="\n">
<globals; separator="\n">
<functions; separator="\n">
<mainfunctions; separator="\n">
>>
extended(e,p)::="<if(e)><e> <p> <else><p><endif>"
setident(id)::=<<<id; separator=",">
>>
setLib(lib)::="#include <lib>"
setFile(name)::="<name>"
getTypedeclarator(list)::=<< <list; separator=",">
>>
mainfunct(arg,locals,stats,fr)::=<<
int main()
{
  <locals; separator="\n">
  <stats; separator="\n">
  return 0;
}
<if(fr)><fr><endif>
>>

pointer(a)::="<a:{n|<n>}>"
castexprs(t,p)::="(<t> <p>)"

typecastexpr(c,id,f,e)::=<<(<c>)<if(id)><id><elseif(f)><f><elseif(
e)>(<e>)<endif>
>>
selHead(id)::="(<id>)"
selBody(p)::="(<p>)"
getPE(id,fc,pe,ce,dot)::=<<
<if(id)><id><else> <fc><if(pe)> <pe><if(ce)>
<ce><else><dot><endif><endif><endif>
>>
mdef(p,r)::="#define <if(!r)><p><else><p> <r><endif>"
mifndef(r)::="#ifndef <r>"
mendif()::="#endif"
melse()::="#else"
getArg(v,s)::="(<v><s>)"
nameMacro(id,type)::="<id><type>"
setMacro(id,expr)::="#define<id> <expr>"
struct(point,name,idx)::=<<
<if(idx)>struct <pointer(point)> <name>{
   <struktur(idx)>
};<endif>
>>
getName(name)::="<typeName(name)>"
typeName(id)::=<< <id;separator=",">
>>
getArrayForm(idx)::=<<
<if(idx)><idx;separator="]["><else>[]<endif>
>>
getEndArray(startArray,endArray)::=<<
<startArray,endArray:{s,e|<e>}>
>>
variable(v,point,type,name,idx) ::= <<
```

```
<if(!type)><if(!v)>struct <endif><pointer(point)> <name><if(idx)>{
<idx>
};<endif><endif>
<if(type)><type> <pointer(point)>
<name><if(idx)>[<idx>]<endif>;<endif>
>>
initVar(point,type,name,idx,init)::="<type> <pointer(point)>
<name><if(idx)>[<idx>]<endif> <if(init)> <assigninit(init)> ;
<else> ; <endif>"
assigninit(init)::="= <init>"
constant(pointer,type,name,idx) ::="const <type> <pointer>
<name><if(idx)>[<idx>]<endif>;"
initConst(pointer,type,name,idx,init)::="const <type> <pointer>
<name><if(idx)>[<idx>]<endif> <if(init)> <assigninit(init)> ;
<else> ; <endif>"

initarray(type,name,size,init)::="<type> <name>[<size>] <if(init)>
<assignarray(init)> ; <endif> ; <endif>"
assignarray(init)::= "= {<init>}"
function(type,funcType,name,args,locals,stats,s,p) ::= <<
<if(s)><funcType> <type> <if(p)><p> <endif> <name>(<args ;
separator=",">)<s><else>
<funcType> <type> <if(p)><p> <endif> <name>(<args ;
separator=",">) {
    <locals; separator="\n">
    <stats; separator="\n">
}<endif>
>>

freal(type,funcType,name,args,locals,stats,p) ::= <<
<funcType> <type> <if(p)><p> <endif> <name>(<args ;
separator=",">){
    <locals; separator="\n">
    <stats; separator="\n">
}<endif>
>>
declType(vars,point,atype,btype,sform,eform,uform)::=<<
typedef <if(vars)> <atype> <point> <vars>;
<elseif(btype)><atype> <vars>[<getArrayForm(btype)>];
<elseif(sform)><sform>
<elseif(eform)><eform>
<elseif(uform)><uform><endif><endif><endif><endif><endif>
>>
getTagID(id,p)::="<if(id)><id><p><endif>"
setEnumType(et,ef,id)::=<<<et> <ef> <if(id)><id><endif>;
>>
enumTag(key,tag)::="<key> <if(tag)><tag><endif>"
getenumVal(v)::=<<
{<v;separator=",">}
>>
setStructType(st,sf,id)::=<<<st> <sf> <if(id)><id><endif>;
>>
structTag(key,tag)::="<key> <if(tag)><tag><endif>"
struktur(field)::=<<
{
  <getVar(field)>
}
>>
setUnionType(ut,uf,id)::=<<<ut> <uf> <if(id)><id><endif>;
```

```
>>
unionTag(key,tag)::="<key> <if(tag)><tag><endif>"
union(field)::=<<
{
  <getVar(field)>
}
>>
getVar(field)::=<< <field;separator="\n">
>>

structForm(cont)::= <<
{
  <cont>
}
>>
constEnum(id,d,c,f)::="<id> = <if(d)><d><elseif(c)><c><else><f>"
tagName(tag)::=<< <tag;separator=",">
>>

getTag(tag)::="<tagName(tag)>"
getenum(e)::=<<
<e;separator=",">
>>
getenumfield(f)::=<<<f;separator=",">
>>
inputstat()::="scanf"
outputstat()::="printf"
type_int() ::= "int"
type_char() ::= "char"
type_float() ::= "float"
type_pointer()::= "*"
type_string()::= "char *"
type_void()::= "void"
type_short()::= "short"
type_long()::= "long"
type_double()::= "double"
type_signed()::= "signed"
type_unsigned()::= "unsigned"
type_user_object(name)::="<name>"
castid(c,t)::="(<c>)<t>"
postfix(opr,term)::="<if(opr)><term><opr><else><term><endif>"
negasi(opr1,term)::="<if(opr1)><opr1><endif><term>"
unary(opr1,unary)::="<if(opr1)><opr1><endif><unary>"

mult(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif>"
add(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif>"
relation(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif
>"
referensi(op1,opr,op2)::="<op1><if(opr)><opr><op2><endif>"
shiftbit(op1,opr,op2)::="<if(opr)><op1><opr><op2><else><op1><endif
>"
exprs(op1,opr,op2)::=<<
<if(opr)><op1> <opr> <op2><else><op1><endif>
>>

assignEq()::="="
parameter(a,type,tinout,name) ::=<<
<name:{n|<type> <tinout> <n> <a>};separator=", ">
>>
```

```
paramPE(pe)::=<<<pe:{n|<n>};separator=",">
>>
paramE(ap)::=<<<ap:{n|<n>};separator=",">
>>
fterm(f)::="<f>"
procedurecall(f)::="<f>;"
functioncall(f,ap)::="<if(ap)><f>(<ap>)<else><f>()<endif>"
getfParam(p)::=<< <p;separator=",">
>>

statProc(p)::="<p>;"
getAP(f,r)::="<f><if(r)><r><endif>"
getAParam(p)::=<< <p;separator=",">
>>
statement(expr) ::="<expr>;"
statementList(locals,stats) ::= <<
{
    <locals; separator="\n">
    <stats; separator="\n">
}
>>
statIO(tag,arg,exp,cp)::=<<
<tag>(<arg><if(exp)>,<exp><endif>);
>>
//for(<id>=<start>;<if(!range)><id>\>=<end>;<if(step)><id>=<id>-
<step><else><id>--
<endif><endif><if(range)><id>\<=<end>;<if(step)><id>=<id>+<step><e
lse><id>++<endif><endif>)
//range=true=Inc
forLoop(id,range,start,end,step)::= <<
for(<if(id)><id>=<start>;<if(range)><if(!step)><end>;<id>++)<else>
<end>;<step>)<endif><else><if(!step)><end>;<id>--
)<else><end>;<step>)<endif><endif><else>;;)<endif>
{
    <locals; separator="\n">
    <stats; separator="\n">
}//endfor
>>
ifstat(log,t,e2)::= <<if(<log>)<t><if(e2)><e2><endif>
>>
getthenstat(s)::="<s>"
getelsestat(s)::="else <s>"
switchstat(id,spart,elseswitch)::=<<
switch(<id>){
  <spart;separator="\n">
  <if(elseswitch)><elseswitch>
};<else>
};<endif>//endswitch
>>
getSwitchPart(id,stat)::=<<
case <id> : <stat>
>>
getElseswitch(stat)::="default : <stat>"
setbreakcont(bc)::="<bc>;"
whilestat(e)::=<<
while(<if(e)><e><else>1<endif>){
  <locals; separator="\n">
  <stats; separator="\n">
};//endwhile
```

```
>>
repeatstat(e)::=<<
do{
  <locals; separator="\n">
  <stats; separator="\n">
}while(<if(e)><e><else>1<endif>);//enddo
>>
decar(id,ue)::="<id><ue>"
assignref(p,c,d)::= "<p><c><d>"
unary_exprs(a,e,pe)::="<a><if(e)><e><else><pe><endif>"
setDot(id)::=".<id>"
setArrow(id)::="-><id>"
refArrow(a)::="<setArrow(a)>"
CastInc(uop,c)::="<uop> <c>"
sizeExprs(e)::="sizeof<e>"
prime(pe)::="(<pe>)"
postexprs(pe,e,id,dr,et,ce,f)::="<if(e)><pe>[<e>]<elseif(id)><pe>[
<id>]<elseif(dr)><pe><dr><elseif(et)><pe><et><elseif(ce)><pe><ce><
elseif(f)><pe><f><else><pe><endif>"
argExprsList(al)::="<al>"
assign(lval,asg,e,c)::= "<lval><asg><if(c)><c><endif><e>;"
assignLoop(lval,asg,e,c)::= "<lval><asg><if(c)><c><endif><e>"

assignfunc(lhs1,pe,point,asg1,rhs1,s)::=
"<if(!s)><if(pe)><pe><else><lhs1><endif><point> <asg1>
<rhs1><else>;<endif>"
refArrayTerm(id,arr)::="<id><arr>"
refTerm(pe,id)::="<id><pe>"
refVar(id,point,n)::="<id><point><if(n)> = <n><endif>"
declsuff(id,e)::="[<id><e>]"
termdeclsuff(a,d)::="<a><d>"
refVariabel(id,point,ds)::="<if(!point)><id><ds><elseif(!ds)><poin
t><id><else><point><id><ds><endif><endif>"
getParam(id)::=<<
<id;separator=",">
>>
getParamPoint(id,point) ::= <<
<id:{n|<point><n>};separator=",">
>>
iconst(value) ::= "<value>"
cconst(value) ::= "<value>"
sconst(value) ::= "<value>"
sconstpas(value)::="<value>"
fconst(value) ::= "<value>"
globalVariable::=initVar
globalConstant::=initConst
globalStruct::=struct
globaldeclType::=declType

hexconstexpr(hex)::="<hex>"
octconstexpr(oct)::="<oct>"
decconstexpr(dec)::="<dec>"
chrconstexpr(chr)::="<chr>"
strconstexpr(str)::="<str>"
strconstexprpas(str)::="<str>"
floconstexpr(flo)::="<flo>"
ifCond(e,ce)::="<if(ce)><e><ce><else><e><endif>"
condExprs(b,c)::=" ? <b> : <c> "
```

```
castexp(t,c,ue)::="<if(ue)><ue><else><t> <c><endif>"
unaryexp(pe,a,uo,ce,s)::="<if(pe)><pe><elseif(a)><a><elseif(uo)><u
o><ce><else><s><endif>"
unInDec(a,ue)::="<a><ue>"
sizeOF(p,ue,t)::="<if(ue)><p><ue><else><p><t><endif>"
postexp(pe,e,blank,arg,a)::=<<
<if(e)><pe><e><elseif(blank)><pe><blank><elseif(arg)><pe><arg><els
e><pe><a><endif>
>>
refdot(id)::=".<id>"
refpoint(id)::="-\><id>"
innerexpr(e)::="(<e>)"
idp(id,p)::="<if(p)>(<p><id>)<else><id><endif>"
oprunNeg()::="!"
oprunNot()::="~"
oprassignEq()::="="
oprmultdiv()::="/"
oprmultmod()::="%"
oprreleq()::="=="
oprrelneq()::="!="
oprshl()::="\<\<"
oprshr()::="\>\>"
oprsand()::=" & "
oprsor()::=" | "
oprexprsor()::="||"
oprexprsand()::="&&"
oprexprsxor()::=" ^ "
returnStat(id)::="<id>"
pointersymbol()::="*"
advtypetag(a,b)::="<a> <b>"
```

Lampiran 8. Laporan Penggunaan Dana

URAIAN PENGGUNAAN DANA
Dibiayai oleh DIKTI Melalui LP2M dengan No. Kontrak : 009/A.35-02/UDN.09/IX/2013

CATATAN KEUANGAN
PENELITIAN DOSEN PEMULA
TAHUN ANGGARAN 2013

1. Judul : Translator Notasi Algoritmik Untuk Pengajaran Pemrograman Dasar
2. Nama Peneliti : Wijanarto,S.Sos, M.Kom
3. Fakultas/Pusat Studi : Fak. Ilmu Komputer / Teknik Informatika
4. Jumlah Biaya : 14.500.000

Tanggal	Uraian	Penerimaan	Pengeluaran	Saldo	
09 September 2013	Terima Tahap I (70%)		14.500.000		
09 September 2013	Potongan PPN 10 %	Potongan PPn 10 %		922727	13.577.273
09 September 2013	Potongan PPh. Ps. 23 (2 %)	Potongan PPh Ps 23 2 %		184545	13.392.728
09 September 2013	Potongan PPh. Ps. 21 (5/6 %)	PPh Ketua/Aggt Sept-Okt		145000	13.247.728
09 September 2013	Honor Peneliti	Minggu 1 September		500.000	12.747.728
09 September 2013	Voucher Simpati	1 buah		100.000	12.647.728
10 September 2013	Biaya Instalasi Software	3 Aplikasi		150.000	12.497.728
10 September 2013	USB Flashdisk 8 GB	2 buah		200.000	12.297.728
10 September 2013	MMC Camera 8 Gb	1 buah		100.000	12.197.728
10 September 2013	Bahan Bakar	30,76 Liter		200.000	11.997.728
10 September 2013	Sewa Mobil	1 hari		250.000	11.747.728
13 September 2013	Modem CDMA	1 Buah		950.000	10.797.728
13 September 2013	Voucher Paket Data	1 Paket		125.000	10.672.728

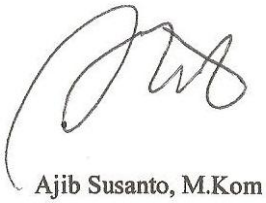
13 September 2013	Buku Folio	1 buah		12.900	10.659.828
13 September 2013	Paket Steadler	1 Paket		14.800	10.645.028
13 September 2013	Ballpoint	2 buah		5.800	10.639.228
13 September 2013	Garisan	2 buah		4.800	10.634.428
14 September 2013	Honor Pengkodean	1 paket		265.000	10.369.428
14 September 2013	Harddisk Ekternal 500 GB	1 buah		425.000	9.944.428
15 September 2013	Honor Diskusi Grup Riset 1	6 Orang		600.000	9.344.428
15 September 2013	Konsumsi Diskusi	1 Paket		376.000	8.968.428
17 September 2013	Honor Asisten Lab	1 orang		50.000	8.918.428
17 September 2013	Konsumsi Laboratorium	5 Buah		112.500	8.805.928
17 September 2013	Sewa Laboratorium Dasar	1 Hari		500.000	8.305.928
24 September 2013	Honor Diskusi Grup Riset 2	3 Orang		300.000	8.005.928
24 September 2013	Konsumsi	1 Paket		266.700	7.739.228
24 September 2013	CDRW	1 box		150.000	7.589.228
25 September 2013	Kertas A4 80 gram	5 rim		180.000	7.409.228
25 September 2013	Kertas F4 70 gram	4 rim		148.000	7.261.228
25 September 2013	Tinta Hitam Ink Jet	1 buah		170.000	7.091.228
25 September 2013	Tinta Warna Ink Jet	1 buah		228.000	6.863.228
03 Oktober 2013	Honor Pelaksana Penulisan Lap Kem	1 orang		200.000	6.663.228
03 Oktober 2013	Voucher Paket Data	1 paket		125.000	6.538.228
03 Oktober 2013	HVS Folio	1 rim		30.000	6.508.228
03 Oktober 2013	fc+Jilid Hard Cover	6 buah		150.000	6.358.228
09 Oktober 2013	Memory 2 GB	1 buah		974.000	5.384.228
09 Oktober 2013	Materai	4 buah		26.000	6.332.228
15 Oktober 2013	Honor pembuatan soal	4 Orang		400.000	4.984.228
15 Oktober 2013	Konsumsi pembuatan soal	1 paket		103.500	6.228.728
15 Oktober 2013	Pembelian Pulsa Paket data	1 Voucher		125.000	4.859.228
22 Oktober 2013	Honor Asisten Treatment Posttest 1	1 Orang		50.000	6.178.728
29 Oktober 2013	Honor Asisten Treatment Posttest 2	1 Orang		50.000	4.809.228

01 Nopember 2013	Honor Peneliti	1 Orang		900.000	5.278.728
09 Nopember 2013	Belanja Pulsa Data Internet	1 Voucher		125.000	6.053.728
12 Nopember 2013	Honor Asisten Treatment Posttest 3	1 Orang		50.000	4.759.228
13 Nopember 2013	Biaya Pengolahan dan Analisis Data	1 Paket		200.000	5.853.728
13 Nopember 2013	Biaya Pengolahan dan Analisis Data	1 Paket		250.000	4.509.228
13 Nopember 2013	Belanja ATK	1 Paket		450.000	5.403.728
14 Nopember 2013	Pembelian Pulsa Paket data	1 Voucher		125.000	4.384.228
14 Nopember 2013	Belanja DVDRW	1 Box		150.000	5.253.728
14 Nopember 2013	Penggandaan Dokumen Teknis	1 Paket		250.000	4.134.228
14 Nopember 2013	Dokumentasi Sosialisasi	1 Paket		250.000	5.003.728
15 Nopember 2013	Honor Peneliti	1 orang		802.828	3.331.400
25 Nopember 2013	Sosialisasi Hasil penelitian sementara	1 paket		150.000	4.853.728
25 Nopember 2013	Honor Asisten Peneliti	3 Orang		150.000	3.181.400
25 Nopember 2013	Honor Ketua Peneliti	1 Orang		200.000	4.653.728
19 Desember 2013	Biaya Prosiding	1 orang		600.000	2.581.400
		Jumlah	14.500.000	14.523.100	23.100

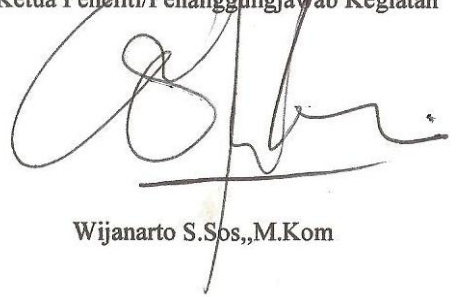
Semarang, 9 Desember 2013

Bendahara Penelitian

Ketua Peneliti/Penanggungjawab Kegiatan



Ajib Susanto, M.Kom



Wijanarto S.Sos,,M.Kom

LAPORAN PENGGUNAAN DANA
PENELITIAN DOSEN PEMULA
TAHUN ANGGARAN 2013

Judul Penelitian : Translator Notasi Algoritmik Untuk Pengajaran Pemrograman Dasar
Nama Peneliti : Wijanarto,S.Sos, M.Kom
Fakultas : Ilmu Komputer
Uang yang diterima :
Tahap 1 : Rp 10.150.000
Tahap 2 : Rp 4.350.000
Jumlah Rp 14.500.000
Penggunaan Rp 14.523.100
Sisa (23.100)

I. Gaji/Honorarium

No.	Nama	Jabatan	Jumlah Minggu	Honorarium per jam	Jumlah
	Honor Peneliti	Ketua Peneliti			500.000
	Honor Asisten Lab	Pelaksana			50.000
	Honor Asisten Treatment Posttest 1	Pelaksana			50.000
	Honor Asisten Treatment Posttest 2	Pelaksana			50.000
	Honor Peneliti	Ketua Peneliti Dan Kelompok Riset			900.000
	Honor Asisten Treatment Posttest 3	Pelaksana			50.000
	Honor Peneliti	Ketua Peneliti Dan Kelompok Riset			802.828
	Honor Asisten Peneliti	Pelaksana			150.000
	Honor Ketua Peneliti	Ketua Peneliti			200.000
	Jumlah gaji/honorarium				**2.752.828**

II. Bahan/Barang Habis Pakai

No.	Jenis Bahan	Volume	Harga Satuan	Jumlah
1	Voucher Simpati	1 paket		100.000
2	Biaya Instalasi Software	3 aplikasi		150.000
3	USB Flashdisk 8 GB	2 buah		200.000
4	MMC Camera 8 Gb	1 buah		100.000
5	Modem CDMA	1 buah		950.000
6	Voucher Paket Data	1 paket		125.000
7	Buku Folio	1 buah		12.900
8	Paket Steadler	1 Paket		14.800
9	Ballpoint	2 buah		5.800
10	Garisan	2 buah		4.800
11	Honor Pengkodean	1 paket		265.000
12	Harddisk Ekternal 500 GB	1 buah		425.000
13	Honor Diskusi Grup Riset 1	6 orang		600.000
14	Konsumsi Diskusi	1 paket		376.000
15	Konsumsi Laboratorium	5 buah		112.500
16	Sewa Laboratorium Dasar	1 hari		500.000
17	Honor Diskusi Grup Riset 2	3 orang		300.000
18	Konsumsi	1 paket		266.700
19	CDRW	1 box		150.000
20	Kertas A4 80 gram	5 rim		180.000
21	Kertas F4 70 gram	4 rim		148.000
22	Tinta Hitam Ink Jet	1 paket		170.000
23	Tinta Warna Ink Jet	1 paket		228.000
24	Honor Pelaksana Penulisan Lap Kem	1 paket		200.000
25	Voucher Paket Data	1 paket		125.000
26	HVS Folio	1 rim		30.000
27	fc+Jilid Hard Cover	6 buah		150.000

28	Memory 2 GB	1 buah		974.000
29	Materai	4 buah		26.000
30	Honor pembuatan soal	4 orang		400.000
31	Konsumsi pembuatan soal	1 paket		103.500
32	Pembelian Pulsa Paket data	1 paket		125.000
33	Belanja Pulsa Data Internet	1 orang		125.000
34	Biaya Pengolahan dan Analisis Data	1 paket		200.000
35	Belanja ATK	1 orang		450.000
36	Pembelian Pulsa Paket data	1 buah		125.000
37	Belanja DVDRW	1 buah		150.000
38	Penggandaan Dokumen Teknis	2 buah		250.000
	Jumlah Bahan/Barang Habis Pakai			**8.818.000**

III. Peralatan dan Lain-lain

No.	Jenis Peralatan	Volume	Harga Satuan	Jumlah
1	Biaya Pengolahan dan Analisis Data	1 paket		250.000
2	Dokumentasi Sosialisasi	1 paket		250.000
3	Sosialisasi Hasil penelitian sementara	1 paket		150.000
4	Biaya Prosiding	1 orang		600.000
	Jumlah Peralatan			**1.250.000**

IV. Perjalanan

No.	Nama yang bepergian	Golongan	Tujuan	Jumlah
1	Sewa Mobil 1 hari (untuk Instalasi dan Reinstalasi Aplikasi 10 September 2013)	Luar Kota	Mranggen-Semarang	250.000
2	Bahan Bakar			200.000
	Jumlah Perjalanan			**450.000**

V. Lain-lain (PAJAK)

No.	Jenis Pengeluaran	Jumlah dalam Kuitansi (Rp.)	Keterangan		Jumlah Pajak (Rp.)
1	Potongan PPh Ps 21 (2%)		5X30 mgg	8.850	184.545
2	Potongan PPh Ps 21 (5/6%)		4x20 mgg	7.375	145.000
4	Potongan PPN 10 %		1 x	2.386.364	922.727
	Jumlah lain-lain (Pajak)				**1.252.272**

VI. Rekapitulasi Penggunaan Biaya

No.	Jenis	Jumlah
I	Gaji/Honorarium	2.752.828
II	Bahan/Barang Habis Pakai	8.818.000
III	Peralatan	1.250.000
IV	Perjalanan	450.000
V	Lain-lain (Pajak)	1.252.272
	Total Biaya	**14.523.100**

Mengetahui,
LPPM Udinus

Y.Tyas Catur P,, S.Si. M.Kom
NPP. 0686.11.1994.046

Semarang, 9-12-2013

Wijanarto, S.Sos, M.Kom
NPP. 0686.11.2009.354

Lampiran 9a Uji Normalitas Kemampuan Kelompok Eksperimen

NPar Tests

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Nilai Posttest KE	38	76.89	10.671	60	93

One-Sample Kolmogorov-Smirnov Test

		Nilai Posttest KE
N		38
Normal Parameters[a,b]	Mean	76.89
	Std. Deviation	10.671
Most Extreme Differences	Absolute	.116
	Positive	.097
	Negative	-.116
Kolmogorov-Smirnov Z		.713
Asymp. Sig. (2-tailed)		.690

a. Test distribution is Normal.

b. Calculated from data.

Lampiran 9b Uji Normalitas Kemampuan Kelompok Kontrol

NPar Tests

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Nilai Posttest KK	38	64.68	12.228	29	85

One-Sample Kolmogorov-Smirnov Test

		Nilai Posttest KK
N		38
Normal Parameters[a,b]	Mean	64.68
	Std. Deviation	12.228
Most Extreme Differences	Absolute	.187
	Positive	.095
	Negative	-.187
Kolmogorov-Smirnov Z		1.153
Asymp. Sig. (2-tailed)		.140

a. Test distribution is Normal.

b. Calculated from data.

Lampiran 9c Uji Normalitas Kecepatan Kelompok Eksperimen

NPar Tests

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Waktu Posttest KE	38	32.03	12.594	17	57

One-Sample Kolmogorov-Smirnov Test

		Waktu Posttest KE
N		38
Normal Parameters[a,b]	Mean	32.03
	Std. Deviation	12.594
Most Extreme Differences	Absolute	.159
	Positive	.159
	Negative	-.128
Kolmogorov-Smirnov Z		.980
Asymp. Sig. (2-tailed)		.293

a. Test distribution is Normal.

b. Calculated from data.

Lampiran 9d Uji Normalitas Kecepatan Kelompok Kontrol

NPar Tests

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
WaktuPosttest	76	38.30	13.138	17	60

One-Sample Kolmogorov-Smirnov Test

		WaktuPosttest
N		76
Normal Parameters[a,b]	Mean	38.30
	Std. Deviation	13.138
Most Extreme Differences	Absolute	.103
	Positive	.094
	Negative	-.103
Kolmogorov-Smirnov Z		.896
Asymp. Sig. (2-tailed)		.398

a. Test distribution is Normal.

b. Calculated from data.

LAMPIRAN 10a Uji T Dan Homogenitas Varian Kemampuan

T-Test

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
NilaiPostTest	Kelompok Eksperimen	38	76.89	10.671	1.731
	Kelompok Kontrol	38	64.68	12.228	1.984

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
NilaiPostTest	Equal variances assumed	.302	.584	4.638	74	.000	12.211	2.633
	Equal variances not assumed			4.638	72.670	.000	12.211	2.633

Independent Samples Test

		t-test for Equality of Means	
		99% Confidence Interval of the Difference	
		Lower	Upper
NilaiPostTest	Equal variances assumed	5.250	19.171
	Equal variances not assumed	5.246	19.175

LAMPIRAN 10b Uji T Dan Homogenitas Varian Kecepatan

T-Test

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
WaktuPosttest	Kelompok Eksperimen	38	32.03	12.594	2.043
	Kelompok Kontrol	38	44.58	10.505	1.704

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means				
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
WaktuPosttest	Equal variances assumed	3.177	.079	-4.718	74	.000	-12.553	2.661
	Equal variances not assumed			-4.718	71.692	.000	-12.553	2.661

Independent Samples Test

		t-test for Equality of Means	
		99% Confidence Interval of the Difference	
		Lower	Upper
WaktuPosttest	Equal variances assumed	-19.587	-5.518
	Equal variances not assumed	-19.593	-5.512

LAMPIRAN 11a Frekuensi dan Histogram Kemampuan Kelompok Eksperimen

Frequencies

Statistics

		Posttest	Interval Nilai Posttest KE
N	Valid	38	53
	Missing	15	0
Mean		76.89	
Std. Error of Mean		1.731	
Median		79.50	
Mode		61[a]	
Std. Deviation		10.671	
Variance		113.881	
Range		33	
Minimum		60	
Maximum		93	
Sum		2922	

a. Multiple modes exist. The smallest value is shown

Frequency Table

Posttest

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	60	1	1.9	2.6	2.6
	61	3	5.7	7.9	10.5
	62	1	1.9	2.6	13.2
	63	1	1.9	2.6	15.8
	64	2	3.8	5.3	21.1
	66	1	1.9	2.6	23.7
	67	1	1.9	2.6	26.3
	68	1	1.9	2.6	28.9
	69	1	1.9	2.6	31.6
	71	1	1.9	2.6	34.2
	72	2	3.8	5.3	39.5
	74	2	3.8	5.3	44.7
	77	1	1.9	2.6	47.4
	79	1	1.9	2.6	50.0
	80	2	3.8	5.3	55.3
	81	3	5.7	7.9	63.2

	84	2	3.8	5.3	68.4
	85	2	3.8	5.3	73.7
	86	1	1.9	2.6	76.3
	87	1	1.9	2.6	78.9
	89	3	5.7	7.9	86.8

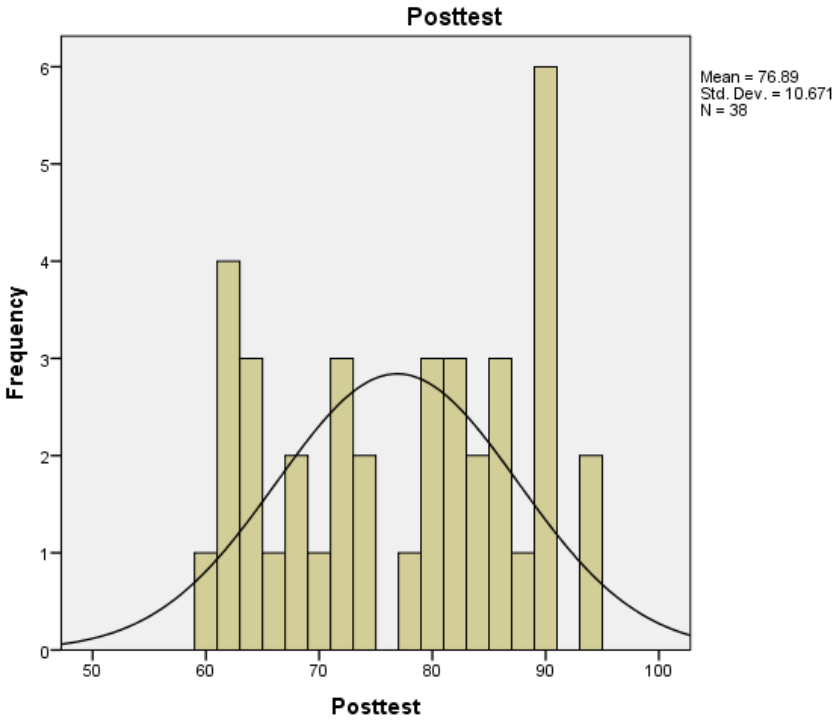
Posttest

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	90	3	5.7	7.9	94.7
	93	2	3.8	5.3	100.0
	Total	38	71.7	100.0	
Missing	System	15	28.3		
Total		53	100.0		

Interval Nilai Posttest KE

		Frequency	Percent	Valid Percent	Cumulative Percent
		15	28.3	28.3	28.3
	60 - 69.99	12	22.6	22.6	50.9
Valid	70 - 84.99	14	26.4	26.4	77.4
	85 - 100	12	22.6	22.6	100.0
	Total	53	100.0	100.0	

Histogram



LAMPIRAN 11b Frekuensi dan Histogram Kemampuan Kelompok Kontrol

Frequencies

Statistics

		PostTest	Interval Nilai Posttest KK
N	Valid	38	53
	Missing	15	0
Mean		64.68	
Std. Error of Mean		1.984	
Median		64.00	
Mode		61	
Std. Deviation		12.228	
Variance		149.519	
Range		56	
Minimum		29	
Maximum		85	
Sum		2458	

Frequency Table

PostTest

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	29	1	1.9	2.6	2.6
	30	1	1.9	2.6	5.3
	50	2	3.8	5.3	10.5
	58	2	3.8	5.3	15.8
	59	2	3.8	5.3	21.1
	60	2	3.8	5.3	26.3
	61	5	9.4	13.2	39.5
	62	2	3.8	5.3	44.7
	63	1	1.9	2.6	47.4
	64	3	5.7	7.9	55.3
	65	2	3.8	5.3	60.5
	66	1	1.9	2.6	63.2
	68	1	1.9	2.6	65.8
	69	1	1.9	2.6	68.4
	70	3	5.7	7.9	76.3
	72	1	1.9	2.6	78.9
	75	1	1.9	2.6	81.6

	78	2	3.8	5.3	86.8
	79	1	1.9	2.6	89.5
	81	1	1.9	2.6	92.1
	85	3	5.7	7.9	100.0

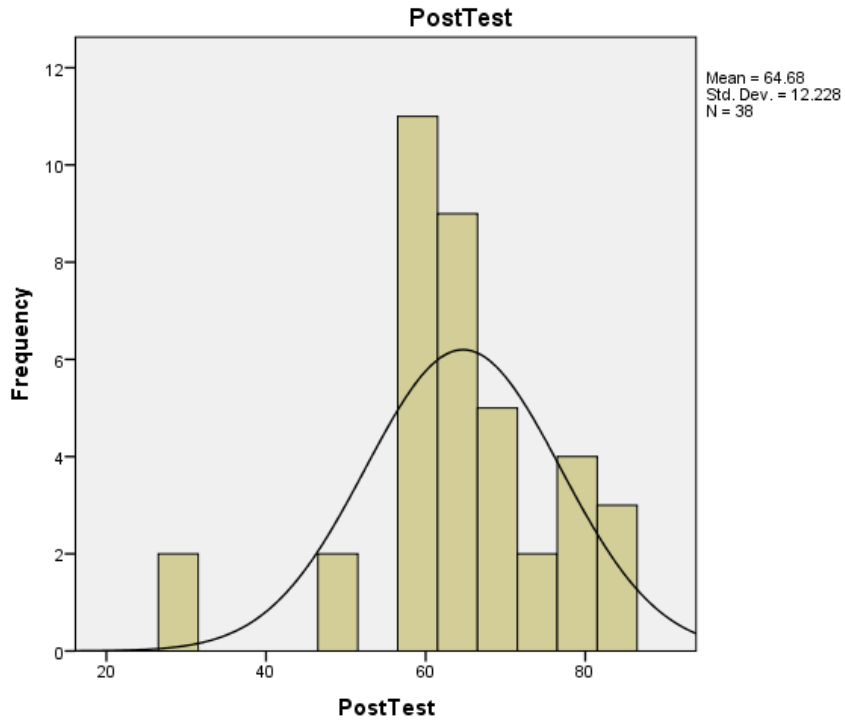
PostTest

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Total	38	71.7	100.0	
Missing	System	15	28.3		
Total		53	100.0		

Interval Nilai Posttest KK

		Frequency	Percent	Valid Percent	Cumulative Percent
		15	28.3	28.3	28.3
	0 - 49.99	2	3.8	3.8	32.1
	50 - 59.99	6	11.3	11.3	43.4
Valid	60 - 69.99	18	34.0	34.0	77.4
	70 - 84.99	9	17.0	17.0	94.3
	85 - 100	3	5.7	5.7	100.0
	Total	53	100.0	100.0	

Histogram



LAMPIRAN 11c Frekuensi dan Histogram Kecepatan Kelompok Eksperimen

Frequencies

Statistics

		Waktu Posttest KE	Interval Waktu Posttest KE
N	Valid	38	53
	Missing	15	0
Mean		32.03	
Std. Error of Mean		2.043	
Median		29.00	
Mode		17	
Std. Deviation		12.594	
Variance		158.621	
Range		40	
Minimum		17	
Maximum		57	
Sum		1217	

Frequency Table

Waktu Posttest KE

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	17	4	7.5	10.5	10.5
	18	1	1.9	2.6	13.2
	19	2	3.8	5.3	18.4
	20	2	3.8	5.3	23.7
	21	2	3.8	5.3	28.9
	22	2	3.8	5.3	34.2
	23	1	1.9	2.6	36.8
	25	3	5.7	7.9	44.7
	27	1	1.9	2.6	47.4
	28	1	1.9	2.6	50.0
	30	3	5.7	7.9	57.9
	34	1	1.9	2.6	60.5
	35	1	1.9	2.6	63.2
	40	1	1.9	2.6	65.8
	42	3	5.7	7.9	73.7
	45	2	3.8	5.3	78.9
	46	2	3.8	5.3	84.2

	47	3	5.7	7.9	92.1
	53	1	1.9	2.6	94.7
	56	1	1.9	2.6	97.4
	57	1	1.9	2.6	100.0

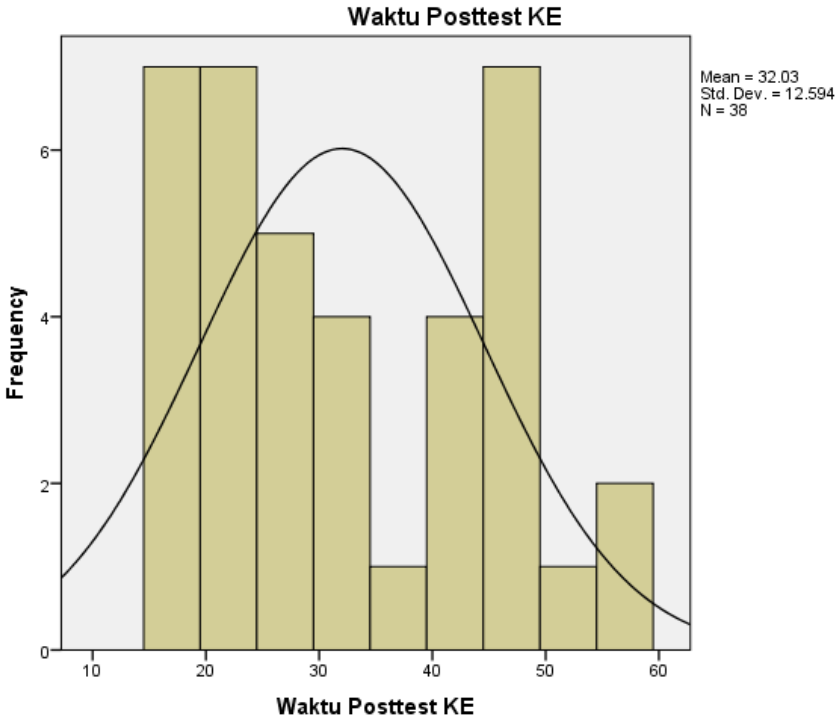
Waktu Posttest KE

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Total	38	71.7	100.0	
Missing	System	15	28.3		
Total		53	100.0		

Interval Waktu Posttest KE

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		15	28.3	28.3	28.3
	0 - 29.99	19	35.8	35.8	64.2
	30 - 39.99	5	9.4	9.4	73.6
	40 - 49.99	11	20.8	20.8	94.3
	50 - 50.99	3	5.7	5.7	100.0
	Total	53	100.0	100.0	

Histogram



LAMPIRAN 11d Frekuensi dan Histogram Kecepatan Kelompok Kontrol

Frequencies

Statistics

		Interval Waktu Posttest KK	Waktu Posttest KK
N	Valid	53	38
	Missing	0	15
Mean			44.58
Std. Error of Mean			1.704
Median			46.33[a]
Mode			56
Std. Deviation			10.505
Variance			110.358
Range			39
Minimum			21
Maximum			60
Sum			1694

a. Calculated from grouped data.

Frequency Table

Interval Waktu Posttest KK

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		15	28.3	28.3	28.3
	0 - 29.99	3	5.7	5.7	34.0
	30 - 39.99	11	20.8	20.8	54.7
	40 - 49.99	10	18.9	18.9	73.6
	50 - 59.99	13	24.5	24.5	98.1
	60 - 69.99	1	1.9	1.9	100.0
	Total	53	100.0	100.0	

Waktu Posttest KK

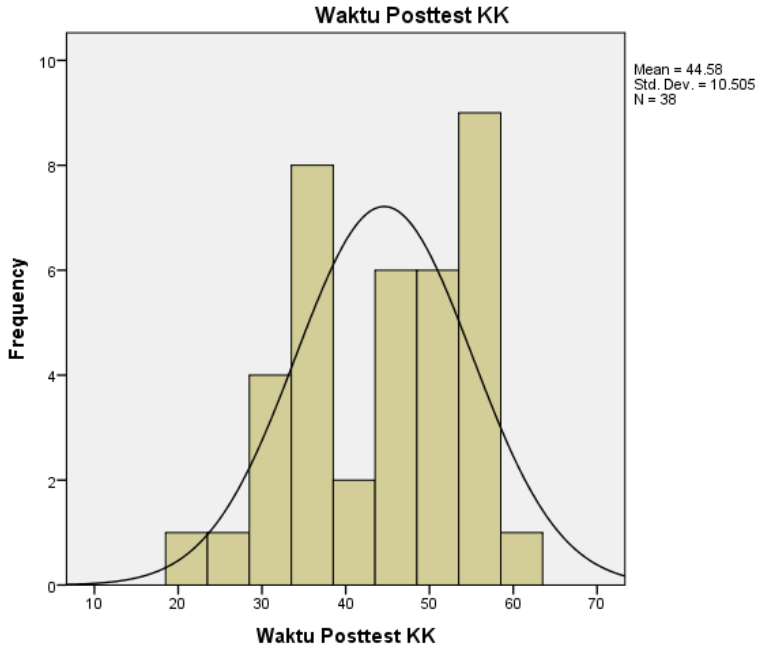
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	21	1	1.9	2.6	2.6
	25	1	1.9	2.6	5.3
	29	1	1.9	2.6	7.9
	30	1	1.9	2.6	10.5
	32	1	1.9	2.6	13.2
	33	1	1.9	2.6	15.8
	34	1	1.9	2.6	18.4
	35	2	3.8	5.3	23.7

	36	1	1.9	2.6	26.3
	37	2	3.8	5.3	31.6
	38	2	3.8	5.3	36.8
	42	1	1.9	2.6	39.5
	43	1	1.9	2.6	42.1
	44	1	1.9	2.6	44.7
	45	1	1.9	2.6	47.4
	46	1	1.9	2.6	50.0
	47	2	3.8	5.3	55.3
	48	1	1.9	2.6	57.9
	49	2	3.8	5.3	63.2
	50	3	5.7	7.9	71.1
	52	1	1.9	2.6	73.7

Waktu Posttest KK

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	56	4	7.5	10.5	84.2
	57	2	3.8	5.3	89.5
	58	3	5.7	7.9	97.4
	60	1	1.9	2.6	100.0
	Total	38	71.7	100.0	
Missing	System	15	28.3		
Total		53	100.0		

Histogram



LAMPIRAN 12 Uji Instrumen Soal Pemrograman Dasar

No	Mahasiswa	Skor Soal Pemrograman				
		Prototype	Algoritma	Validitas Output	Waktu	Total
1	M1	5.0	4.0	4.0	4.0	17
2	M 2	4.0	4.0	5.0	4.0	17
3	M 3	4.0	5.0	4.0	5.0	18
4	M 4	5.0	5.0	5.0	4.0	19
5	M 5	4.0	4.0	4.0	5.0	17
6	M 6	5.0	4.0	4.0	5.0	18
7	M 7	5.0	5.0	5.0	4.0	19
8	v8	4.0	5.0	4.0	4.0	17
9	M 9	4.0	4.0	4.0	5.0	17
10	M 10	4.0	4.0	4.0	4.0	16
11	M 11	4.0	5.0	5.0	4.0	18
12	M 12	5.0	4.0	5.0	5.0	19
13	M 13	5.0	4.0	4.0	5.0	18
14	M 14	4.0	5.0	5.0	5.0	19
15	M 15	5.0	5.0	4.0	5.0	19
16	M 16	3.0	3.0	4.0	4.0	14
17	M 17	4.0	4.0	4.0	4.0	16
18	M 18	5.0	4.0	5.0	4.0	18
19	M 19	5.0	5.0	4.0	4.0	18
20	M 20	4.0	4.0	5.0	4.0	17
21	M 21	4.0	4.0	4.0	5.0	17
22	M 22	4.0	5.0	4.0	4.0	17
23	M 23	3.0	4.0	4.0	4.0	15
24	M 24	5.0	4.0	4.0	5.0	18
25	M 25	5.0	4.0	4.0	4.0	17
26	M 26	4.0	5.0	5.0	5.0	19
27	M 27	4.0	5.0	4.0	5.0	18
28	M 28	5.0	5.0	5.0	5.0	20
29	M 29	2.0	2.0	2.0	3.0	9
30	M 30	4.0	5.0	5.0	4.0	18
31	M 31	5.0	4.0	4.0	4.0	17
32	M 32	4.0	5.0	4.0	4.0	17
33	M 33	4.0	4.0	4.0	4.0	16
34	M 34	5.0	4.0	4.0	4.0	17
35	M 35	4.0	4.0	4.0	4.0	16
36	M 36	5.0	5.0	4.0	4.0	18
37	M 37	5.0	5.0	4.0	4.0	18
38	M 38	5.0	5.0	4.0	4.0	18

LAMPIRAN 12 Uji Instrumen Soal Pemrograman Dasar

No	Mahasiswa	Skor Soal Pemrograman				
		Prototype	Algoritma	Validitas Output	Waktu	Total
1	M1	5.0	4.0	4.0	4.0	17
2	M 2	4.0	4.0	5.0	4.0	17
3	M 3	4.0	5.0	4.0	5.0	18
4	M 4	5.0	5.0	5.0	4.0	19
5	M 5	4.0	4.0	4.0	5.0	17
6	M 6	5.0	4.0	4.0	5.0	18
7	M 7	5.0	5.0	5.0	4.0	19
8	v8	4.0	5.0	4.0	4.0	17
9	M 9	4.0	4.0	4.0	5.0	17
10	M 10	4.0	4.0	4.0	4.0	16
11	M 11	4.0	5.0	5.0	4.0	18
12	M 12	5.0	4.0	5.0	5.0	19
13	M 13	5.0	4.0	4.0	5.0	18
14	M 14	4.0	5.0	5.0	5.0	19
15	M 15	5.0	5.0	4.0	5.0	19
16	M 16	3.0	3.0	4.0	4.0	14
17	M 17	4.0	4.0	4.0	4.0	16
18	M 18	5.0	4.0	5.0	4.0	18
19	M 19	5.0	5.0	4.0	4.0	18
20	M 20	4.0	4.0	5.0	4.0	17
21	M 21	4.0	4.0	4.0	5.0	17
22	M 22	4.0	5.0	4.0	4.0	17
23	M 23	3.0	4.0	4.0	4.0	15
24	M 24	5.0	4.0	4.0	5.0	18
25	M 25	5.0	4.0	4.0	4.0	17
26	M 26	4.0	5.0	5.0	5.0	19
27	M 27	4.0	5.0	4.0	5.0	18
28	M 28	5.0	5.0	5.0	5.0	20
29	M 29	2.0	2.0	2.0	3.0	9
30	M 30	4.0	5.0	5.0	4.0	18
31	M 31	5.0	4.0	4.0	4.0	17
32	M 32	4.0	5.0	4.0	4.0	17
33	M 33	4.0	4.0	4.0	4.0	16
34	M 34	5.0	4.0	4.0	4.0	17
35	M 35	4.0	4.0	4.0	4.0	16
36	M 36	5.0	5.0	4.0	4.0	18
37	M 37	5.0	5.0	4.0	4.0	18
38	M 38	5.0	5.0	4.0	4.0	18