IMPLEMENTASI SISTEM DETEKSI KESALAHAN SOURCE CODE C++ MENGGUNAKAN TEKNIK KOMPILASI

Imay Kurniawan[1]

[1] Teknik Informatika STT Wastukancana Purwakarta
[1]imaykurniawan2013@gmail.com

Abstrak

Praktikum pemrograman C++ sebagai salah satu media pengukur tingkat pemahaman mahasiswa dalam hal membuat program C++. Banyaknya *source code* yang harus diperiksa oleh Dosen mengakibatkan sulitnya melakukan pemeriksaan masing-masing tugas milik mahasiswa. Analisis leksikal dan analisis sintak merupakan tahapan dari teknik kompilasi. Analisis leksikal dan sintak digunakan untuk mendeteksi kesalahan source code yang disebabkan karena kesalahan leksikal dan kesalahan sintak. Sistem deteksi kesalahan source code C++ dapat digunakan untuk mendeteksi kesalahan source code yang disebabkan karena kesalahan leksikal dan sintak. Proses pendeteksi kesalahan source code mengacu kepada daftar token yang sudah dikelompokkan dengan analisis leksikal. Hasil deteksi kesalahan source code menampilkan daftar kesalahan source code dan skor nilai tugas mahasiswa.

Kata kunci : Kesalahan s*ource Code, Analisis leksikal,analisis sintak,C++,Token*

1. Pendahuluan

1.1. Latar Belakang

Laboratorium praktikum pemrograman C++ Prodi Teknik Infomatika Sekolah Tinggi Teknologi Wastukancana sebagai tempat pembelajaran bagi para mahasiswa informatika yang mengikuti kelas Algoritma dan Pemrograman. Praktikum pemrograman sebagai salah satu media pengukur tingkat pemahaman mahasiswa dalam hal membuat program C++.

Permasalahan yang terjadi, banyaknya *source code tugas mahasiswa* yang harus diperiksa oleh Dosen mengakibatkan sulitnya melakukan pemeriksaan. Pemeriksaan yang dilakukan untuk mengetahui apakah ada kesalahan source code di dalam tugas mahasiswa. Kesalahan source code terdiri dari tiga kesalahan yaitu kesalahan leksikal,sintaks, dan semantik.

Pemeriksaan kesalahan source code dapat menggunakan Teknik kompilasi yang terdiri tiga tahap, yaitu tahap analisis leksikal, analisis sintaks, dan analisis semantik. Dalam penelitian ini yang dibahas hanya tahap analisis leksikal dan analisis sintak. Analisis leksikal ini meliputi penguraian *source code* dari program sumber kedalam jenis token
identifier,keyword,delimeter,operator,konstanta string, dan konstanta numerik. Kemudian dilakukan pemeriksaan *source code* yang telah menjadi token. Jika ditemukan token yang tidak sesuai maka program akan mencatatnya sebagai suatu kesalahan. Setelah semua token diperiksa selanjutnya dilakukan perhitungan nilai tugas

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka rumusan masalah yang diambil peneliti adalah sebagai berikut :

1. Bagaimana merancang perangkat lunak sistem deteksi kesalahan source code C++ ?
2. Bagaimana membangun perangkat lunak sistem deteksi kesalahan source code C++ ?

1.3. Tujuan Penelitian

Adapun tujuan dari penulisan penelitian ini adalah sebagai berikut :

1. Peneliti dapat merancang perangkat lunak sistem deteksi kesalahan source code C++
2. Peneliti dapat membangun perangkat lunak sistem deteksi kesalahan source code C++

1.4. Batasan Masalah

Agar pembahasan tidak keluar dari pokok permasalahan, maka permasalahan yang dibahas dalam penulisan penelitian ini sebagai berikut :

1. Token keyword yang diperiksa dibatasi :if,else,for,while,break,continue,do.
2. Model penggambaran sistem menggunakan *Unified Modeling Language (UML).*

2. TINJAUAN PUSTAKA

2.1. Teknik Kompilasi

Menurut Utdirartatmo (200), Teknik kompilasi merupakan teknik dalam melakukan pembacaan suatu program yang ditulis dalam bahasa sumber, kemudian diterjemahkan ke dalam suatu bahasa lain yang disebut bahasa sasaran. Dalam melakukan proses penerjemahan tersebut, sudah barang tentu kompilator akan melaporkan adanya keanehan-keanehan atau kesalahan yang mungkin ditemukannya. Proses kompilasi dari suatu kompilator pada dasarnya dapat dibagi ke dalam 2 bagian utama yaitu bagian analisis dan bagian sintesis. Secara umum proses dalam tahap analis terdiri dari 3 bagian utama, yaitu :

1. Proses analisis leksikal
2. Proses analisis sintaktik
3. Proses analisis semantik

2.2. Analisis Leksikal

Menurut Utdirartatmo(2005),analisis leksikal merupakan fungsi analisis dalam *compiler* bertugas mendekomposisi program sumber menjadi bagian-bagian kecil (*token*). Analisis leksikal atau *scanner* bertugas mengidentifikasi semua besaran pembangun bahasa (leksikal) yang ada pada *source code*. *Scanner* menerima masukan *source code* berupa serangkaian karakter kemudian memilah-milahnya menjadi bagian-bagian kecil yang mempunyai satu arti yang disebut *token*, seperti : *konstanta*, nama variabel, *keyword*, *operator*. *Token-token* ini akan menjadi masukan bagi analisis selanjutnya yaitu analisis *sintaksis*. Dari fungsi *scanner* secara umum seperti telah disebutkan di atas, maka tugas *scanner* secara rinci adalah:

1. Membaca serangkaian karakter dari *source code*.
2. Mengenalinya ke dalam satuan leksikal.
3. Mengubahnya menjadi *token* dan menentukan jenis *token*nya.
4. Mengirimkan *token* ke proses analisis selanjutnya, yaitu analisis *sintaksis*.
5. Mengabaikan karakter *white space* (spasi, *enter*, ganti baris, penanda akhir *file*) dan komentar (*remark*) apabila ada di dalam *source code*.
6. Menangani *error*.

2.3. Analisis Sintak

Sintak adalah aturan-aturan bahasa dalam suatu bahasa pemrograman tertentu. Aturan Sintaks bergantung pada bahasa pemrograman masing-masing. Karena masing-masing bahasa pemrograman memiliki bentuk sintaks yang berbeda-beda. Menurut Utdirartatmo(2005), tugas dari analisis sintak untuk memeriksa urutan kemunculan token.

2.4. Jenis Token

Token adalah satuan terkecil dari bahasa sumber yaitu deretan karakter terpendek yang mengandung arti. Token dikelompokan menjadi beberapa jenis sebagai berikut

1, Identifier
2. Konstanta
3. Keyword
4. Operator
5. Delimiter

2.5. Contoh Token dalam Bahasa Pemrograman C++

Tabel 2.1 Jenis Token

N0	Token	Jenis Token
1.	If,else,for,while,break ,continue	keyword
2.	<,>,=,+,*,-,/	operator
3.	(,[,},{,)	delimiter
4.	variabel	identifier
5.	0,1,2,3,4,55,,7,8,9	Konstanta numerik
6.	“C++”	Konstanta string

2.6. Kesalahan Source Code

Kesalahan source code dapat dibedakan menjadi tiga jenis kesaahan, yaitu kesalahan leksikal,kesalahan,sintak, dan kesalahan semantik.

Kesalahan leksikal diantaranya :

1. Kesalahan ada karakter yang bukan huruf dan angka di dalam token identifier.
2. Kesalahan ada karakter angka pada digit pertama token identifier.
3. Kesalahan penulisan nama token keyword.

Kesalahan sintak diantaranya :

1. Kesalahan karena urutan token dari perintah program tidak sesuai
2. Kurung buka ada tetapi atau kurung tutup tidak ada dalam program sumber
3. Kurung siku buka ada tetapi kurung siku tutup tidak ada dalam program sumber
4. Kurawal buka ada tetapi kurawal tutup tidak ada dalam program sumber

2.7. Rekayasa Perangkat Lunak

Istilah Rekayasa Perangkat Lunak *(RPL)* secara umum disepakati sebagai terjemahan dari istilah *Software Engineering*. Istilah *Software Engineering* mulai dipopulerkan tahun 1968 pada *Software Engineering Conference* yang diselenggarakan oleh NATO. Sebagian orang

mengartikan *RPL* hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak (*software*) dan program komputer.

2.8. *Unified Modelling Language (UML)*

Menurut Munawar(2005), *Unified Modelling Language (UML)* adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan *UML*, kita dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan para perangkat keras, sistem operasi dan jaringan apapun.

UML sebagai salah satu alat bantu yang sering digunakan di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena *UML* menyediakan bahasa pemodelan *visual* yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain .

2.9. *Use Case* Diagram

Menurut Munawar(2005), *use case* adalah deskripsi sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antar *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai (Munawar, 2005).

2.10. *Activity Diagram*

Menurut Munawar(2005), *activity* diagram adalah teknik untuk menggambarkan logika *procedural* proses bisnis dan aliran kerja dalam banyak kasus. *Activity* diagram mempunyai peran seperti halnya *flowchart* akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak

2.11. *Bahasa Pemrograman C++*

C++ adalah bahasa pemrograman komputer yang di buat oleh Bjarne Stroustrup, yang merupakan perkembangan dari bahasa C dikembangkan di Bong Labs (Dennis Ritchie) pada awal tahun 1970-an, Bahasa itu diturunkan dari bahasa sebelumnya, yaitu B, Pada awalnya, bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem Unix. Pada perkembangannya, versi ANSI (American National Standart Institute) bahasa pemrograman C menjadi versi dominan, Meskipun versi tersebut sekarang jarang dipakai dalam pengembangan sistem dan jaringan maupun untuk sistem embedded, Bjarne Stroustrup pada Bel labs pertama kali mengembangkan C++ pada awal 1980-an . Untuk mendukung fitur-fitur pada C++, dibangun efisiensi dan sistem support untuk pemrograman tingkat rendah (*low level coding*). Pada C++ ditambahkan konsep-konsep baru seperti class dengan sifat-sifatnya seperti inheritance dan overloading. Salah satu perbedaan yang paling mendasar dengan bahasa C adalah dukungan terhadap konsep pemrograman berorientasi objek

2.11. Black-Box Testing

Black-Box Testing berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan engineers untuk memperoleh set kondisi input yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program (Pressman, 2010). *Black-Box testing* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan perilaku (*behavior*) atau kesalahan kinerja.
5. Inisialisasi dan pemutusan kesalahan.

3. METODE PENELITIAN

Metode pengembangan perangkat lunak menggunakan paradigma pengembangan perangkat lunak waterfall. Menurut Waterfall adalah sebuah pengembangan model perangkat lunak yang dilakukan secara berurutan atau sekuensial, adapun tahapan model ini sebagai berikut: 1) *Analysis*, 2) *Design*, 3) *Coding*, dan 4) *Testing*

3.1. *Tahapan Sistem Deteksi Kesalahan Source Code C++*

1. *Menguraikan isi surce code file program C++ ke dalam kelompok jenis token menggunakan analisis leksikal.*
2. *Mendeteksi kesalahan leksikal menggunakan analisis leksikal .*
3. Mendeteksi kesalahan sintak menggunakan analisis sintak
4. Menghitung skor kesalahan source code

3.2. *Analisis Kebutuhan Non Fungsional*

Analisis kebutuhan non fungsional ini meliputi analisis kebutuhan perangkat keras (hardware), dan perangkat lunak (software) yang digunakan dalam membangun sistem.

3.2.1 Analisis Kebutuhan Perangkat Keras (Hardware)

Spesifikasi perangkat keras yang dibutuhkan antara lain sebagai berikut :

1. Prosesor Intel atom 1,8GHz
2. Chipset Intel
3. Grafis Intel HD Graphics 5500
4. Memori RAM 2GB
5. Storage hard disk 300GB 5400 rpm

3.2.2. Analisis Kebutuhan Perangkat Lunak (Software)

Spesifikasi perangkat lunak yang dibutuhkan antara lain sebagai berikut :

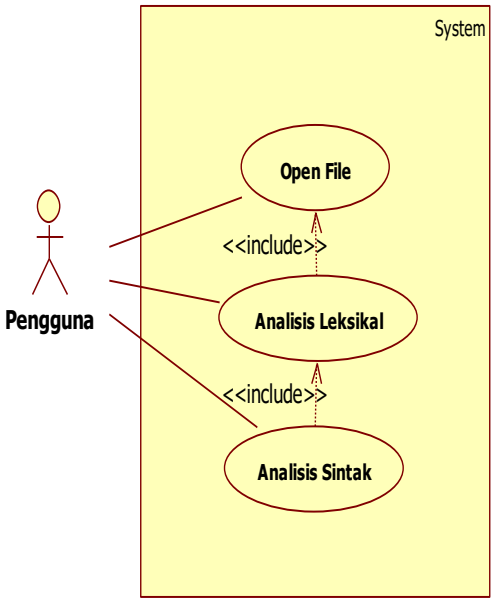
1. Sistem operasi Microsoft *Windows 7 Enterprise 32bit*,
2. Bahasa pemrograman PHP

3.3. Desain Sistem

3.3.1. Desain UML (*Unified Modelling Language*)

3.3.1.1. Use Case Diagram

Dalam tahap ini akan dijelaskan untuk mendeskripsikan apa yang harus dilakukan oleh sistem, digambarkan dalam bentuk *use case* yang bertujuan untuk menunjukkan alur kerja dan proses dari sistem aplikasi yang akan dibuat. *Use Case Diagram* atau diagram *use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat, diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem yang akan dibuat. Proses yang akan digambarkan akan berlangsung secara terstruktur. Berikut merupakan gambaran *use case* diagram untuk sistem yang akan dibangun pada gambar 3.1 :

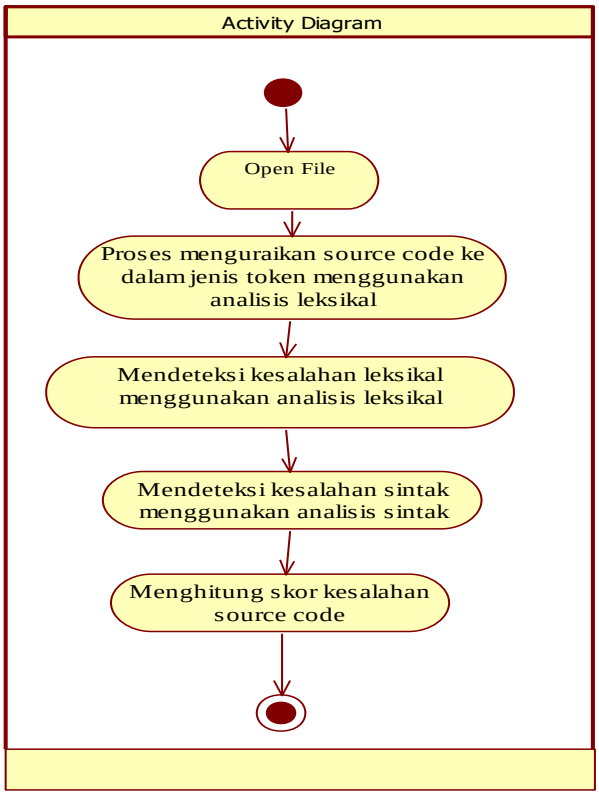


Gambar 3.1. *Use Case Diagram* Sistem

3.3.1.2 Activity Diagram

Activity diagram adalah salah satu cara untuk memodelkan *event-event* yang terjadi dalam suatu *use case*.

Berikut merupakan gambaran *activity* diagram untuk sistem yang akan dibangun pada gambar 3.2 :



Gambar 3.2. *Activity Diagram* Sistem

4. HASIL DAN PEMBAHASAN

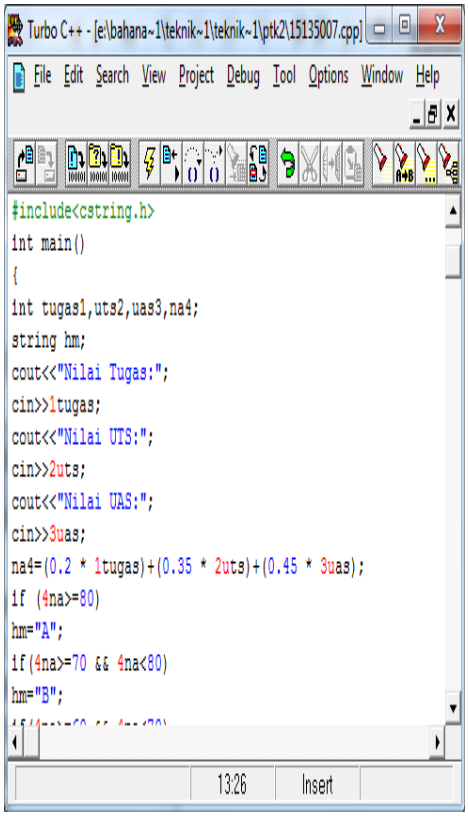
4. 1. Implementasi

Implementasi merupakan kelanjutan dari kegiatan perancangan sistem. Tahap ini merupakan tahap meletakkan sistem supaya siap untuk di operasikan dan dapat dipandang sebagai usaha untuk mewujudkan sistem yang telah di rancang. Pada tahap ini akan diambil tiga sampel tugas mahasiswa yang akan diperiksa kesalahan-kesalahan source code yang berhasil terdeteksi. Hasil pengujian adalah terdeteksi kesalahan source code yang diambil secara random dari tugas mahasiswa.

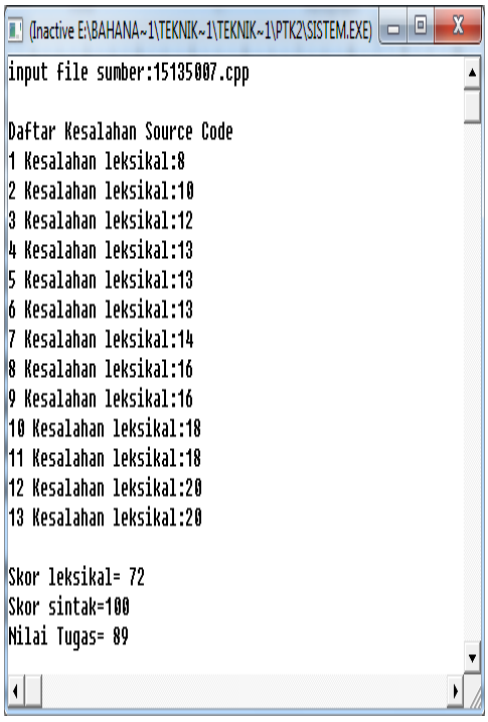
1. File Tugas 15135007

Pada sampel pertama, sistem mendeteksi ada kesalahan source code yang disebabkan

kesalahan leksikal. Isi file source code program sampel pertama dapat dilihat di gambar 4.1, Dengan menggunakan sistem pendeteksi kesalahan source code dapat dideteksi kesalahan source code pada sampel pertama dan diperoleh daftar kesalahan source code dan skor nilai tugas mahasiswa , hasilnya dapat dilihat di gambar 4.2.



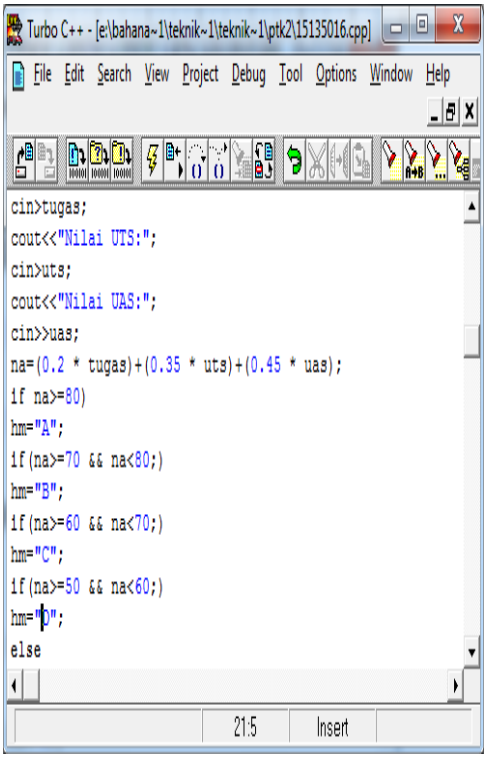
Gambar 4.1. File Sampel 1



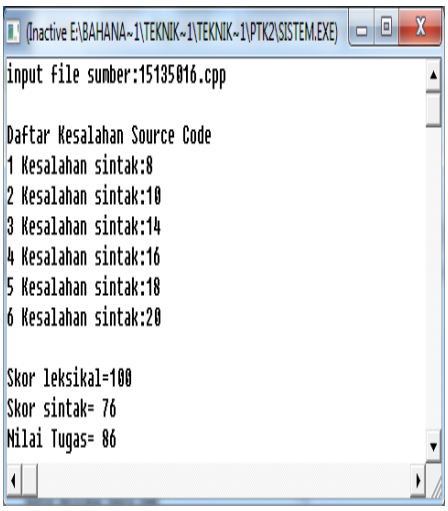
Gambar 4.2. Hasil Deteksi Sampel 1

2. File 15135016

Pada sampel kedua, sistem mendeteksi ada kesalahan source code disebabkan kesalahan sintak. Isi file source code program sampel kedua dapat dilihat di gambar 4.3, Dengan menggunakan sistem pendeteksi kesalahan source code dapat dideteksi kesalahan source code pada sampel kedua dan diperoleh daftar kesalahan source code dan skor nilai tugas mahasiswa , hasilnya dapat dilihat di gambar 4.4.



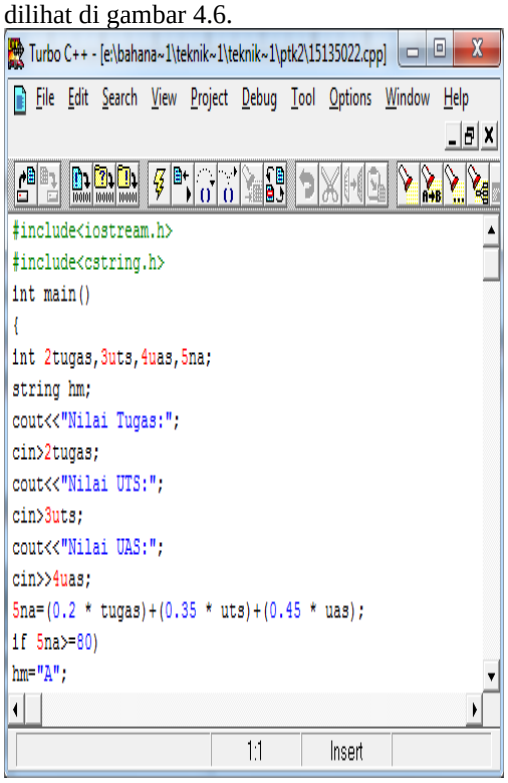
Gambar 4.3. File sampel 2



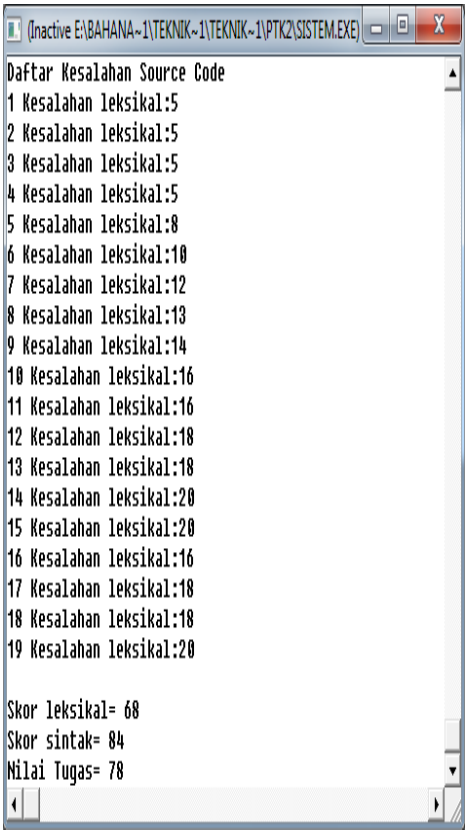
Gambar 4.4. Hasil Deteksi Sampel 2

3. File 15135009 dengan file 15135022

Pada sampel ketiga, sistem mendeteksi ada kesalahan source code disebabkan kesalahan leksikal dan sintak. Isi file source code program sampel ketiga dapat dilihat di gambar 4.5, Dengan menggunakan sistem pendeteksi kesalahan source code dapat dideteksi kesalahan source code pada sampel ketiga dan diperoleh daftar kesalahan source code dan skor nilai tugas mahasiswa , hasilnya dapat dilihat di gambar 4.6.



Gambar 4.3. File sampel 3



Gambar 4.6. Hasil Deteksi Sampel 3

4.2. *Pengujian*

Pengujian yang dilakukan menggunakan teknik pengujian *black box* yang memfokuskan pada domain fungsional dari perangkat lunak. Hasil pengujian dapat dilihat pada tabel 4.2.1

Tabel 4.2.1. Hasil Pengujian

No	Pengujian	Hasil Yang Diharapkan	Keterangan
1	Deteksi kesalahan source code sampel 1	Dapat dideteksi kesalahan source code	Berhasil
2.	Deteksi kesalahan source code sampel 2	Dapat dideteksi kesalahan source code	Berhasil
3.	Deteksi kesalahan source code sampel 3	Dapat dideteksi kesalahan source code	Berhasil

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan tahap penelitian, perancangan, dan tahap implementasi sistem deteksi kesalahan source code dengan menggunakan analisis leksikal, diperoleh kesimpulan sebagai berikut :

1. Analisis leksikal dapat diimplementasikan untuk mendeteksi kesalahan source code dari file source code C++
2. Hasil dari aplikasi ini berupa daftar kesalahan source code C++, dapat dijadikan acuan untuk penilaian tugas source code C++

.

5.2 Saran

Berdasarkan kesimpulan diatas maka dapat megemukakan beberapa saran yang diharapkan dapat menjadi masukan bagi kemajuan sistem yang akan datang.

1. Aplikasi ini mendeteksi kesalahan source code pada bahasa pemrograman C++, untuk kedepannya diharapkan dapat mendeteksi lebih dari satu bahasa pemrograman.

2. Analisis yang digunakan mendeteksi kesalahan source code hanya menggunakan analisis leksikal dan analisis sintak kedepannya dapat ditambah dengan analisis semantik.

DAFTAR PUSTAKA

Bjarne Stroustrup.1977. *The C++ Programming Language*, Pearson Education, Indianapolis

Firrar Utdirartatmo. (2005):Teknik Kompilasi, Graha Ilmu

Henk Alblas, Albert Nymeyer.1996.Practice and prinsiples of compiler building with C,Prentice Hall.

Kadir,A.2010. *Mudah Menjadi Programmer C++*, Penerbit Andi, Yogyakarta.

Munawar, A. 2005. *Pemodelan Visual Dengan UML*,Graha Ilmu,Jakarta.

Michael Sipser. 1997. Introduction to The Theory of Computation, PWS Publishing Company

Pressman, R. S. 2010. *Software engineering a practitioner's Approach*, Palgrave Macmillan.

Robin H.1999.The Essence of Compilers, Prentice-Hall Europe