IMPLEMENTASI SISTEM DETEKSI KEMIRIPAN SOURCE CODE C++ MENGGUNAKAN ANALISIS LEKSIKAL

Imay Kurniawan,M.Kom[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 serta sulitnya mengukur *kredibilitas* masing-masing tugas milik mahasiswa. Analisis Leksikal dapat digunakan untuk mendeteksi kemiripan kode program, dengan proses mengelompokkan kedua program sumber menjadi token identifier,keyword,operator,delimeter, konstanta numerik. Semua token disinpan di dalam array.. *Token* ini digunakan untuk mengidentifikasi struktur program yang akan diperiksa. Selanjutnya melakukan proses pencocokan token dari dua file yang diperiksa. Semakin besar kemiripan alur dan kode program maka semakin besar kemungkinan kode tersebut merupakan *plagiat*. Nilai presentase kemiripan kode program yang dihasilkan dari proses pencocokan token ini yang nantinya digunakan sebagai acuan penentuan *plagiarism.*

Kata kunci : *Source Code, Analisis leksikal,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 serta sulitnya mengukur kredibilitas masing-masing tugas milik mahasiswa. Kode program terperiksa yang memiliki tingkat kemiripan yang cukup tinggi antar kode dapat dijadikan acuan adanya tindakan-tindakan kecurangan seperti melakukan tindakan plagiat kode terhadap tugas mahasiswa lain.

Metode deteksi kemiripan kode menggunakan analisis leksikal dapat digunakan untuk mengubah kode program menjadi *token,* masing-masing kode terperiksa. Semakin besar kemiripan maka semakin besar kemungkinan kode tersebut merupakan hasil plagiat. Sistem pendeteksi kemiripan source code akan melakukan analisis leksikal pada program yang diperiksa dalam hal ini bahasa pemrograman *C++*. Analisis ini meliputi pengubahan *source code* du*a file* tugas mahasiswa kedalam jenis token identifier,,keyword,delimeter,operator,konstanta string, dan konstanta numerik. Kemudian dilakukan pencocokan *source code* yang telah menjadi token yang memiliki kesamaan pada program pembanding berdasarkan struktur yang terbentuk. Jika ditemukan token yang sama maka program akan mencatatnya dan kemudian menghitung tingkaat kemiripan *source code* tersebut dan memberikan nilai presentase kemiripan.

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 kemiripan source code C++ ?
2. Bagaimana membangun perangkat lunak sistem deteksi kemiripan source code C++ ?

1.3. TUJUAN PENELITIAN

Adapun tujuan dari penulisan penelitian ini adalah sebagai berikut :

1. Peneliti dapat merancang perangkat lunak sistem deteksi kemiripan source code C++
2. Peneliti dapat membangun perangkat lunak sistem deteksi kemiripan 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. Bahasa pemrograman C++. Menggunakan Turbo C++ versi 4.5
2. Model penggambaran sistem menggunakan *Unified Modeling Language (UML).*

2. Tinjauan Pustaka

2.1. Pengertian *Plagiarisme*

Plagiarisme merupakan tindakan kriminal yang sering terjadi dalam dunia akademis. *Plagiarisme* itu sendiri berasal dari kata latin "*Plagiarus*" yang berarti penculik dan "*Plagiare*" yang berarti mencuri. Jadi, secara sederhana *plagiat* berarti mengambil ide, kata-kata, dan kalimat seseorang dan memposisikannya sebagai hasil karyanya sendiri atau mengunakan ide, kata-kata, dan kalimat tanpa mencantumkan sumber dimana seorang penulis mengutipnya (Sastroasmoro, 2007).

2.2. Jenis-Jenis *Plagiarisme*

Terdapat jenis jenis *plagiarism*, menurut Sudigdo Sastroasmoro dalam makalahnya berjudul Beberapa Catatan tentang *Plagiarisme* pada tahun 2007, mengklasifikasikan *plagiarisme* dalam empat jenis. Jenis *plagiarisme* berdasarkan klasifikasinya diantaranya:

2.2.1. Jenis *plagiarisme* berdasarkan aspek yang dicuri

Dalam klasifikasi ini terdiri atas 4 kategori yaitu kategori *plagiarisme* ide, *plagiarisme* isi, *plagiarisme* kata, kalimat, paragraf, dan *plagiarisme* total. *Plagiarisme* ide sering dikaitkan dengan penelitian *replikatif*. Penelitian *reflikatif* adalah penelitian yang berdasarkan atas ide orang lain. Apabila dalam melakukan penelitian, analisis yang digunakan sama dengan penelitian sebelumnya sama, maka harus mencantumkan sumber dan alasan ilmiah mengapa penelitian ulang tersebut perlu dilakukan.

2.2.2. Klasifikasi berdasarkan sengaja atau tidaknya *plagiarisme*

Plagiarisme sengaja adalah mencuri atau dengan sengaja menjiplak hasill karya orang lain dengan kepentingannya sendiri. Hal ini dapat terjadi bila seseorang menjiplak ide, kata, frasa, kalimat atau pragraf tanpa mencantumkan sumbernya dan dilakukan dengan sadar untuk kepentingan sendiri.

Plagiarisme tidak disengaja adalah *plagiarisme* yang terjadi karena ketidaktahuan. Ketidaktahuan ini biasa terjadi dalam menggunakan dokumentasi, teknik mengutip karya tulis, dan parafrase kalimat yang keliru. Meskipun plagirisme tidak disengaja namun sanksi yang dikenakan sama seperti *plagiarisme* yang disengaja. Hal tersebut dapat dicegah dengan menunjukan bagaimana menghindari *plagiarisme*.

2.2.3. Klasifikasi berdasarkan proporsi atau persentase kata, kalimat, paragraf yang dibajak

1. *Plagiarisme* ringan, *plagiarisme* yang jumlah poporsi atau persentase kata, kalimat, paragraf yang dibajak tidak melebihi 30 persen (< 30%).
2. *Plagiarisme* sedang, *plagiarisme* yang jumlah poporsi atau persentase kata, kalimat, paragraf yang dibajak antara 30-70 persen.
3. *Plagiarisme* berat, *plagiarisme* yang jumlah poporsi atau persentase kata, kalimat, paragraf yang dibajak lebih dari 70 persen (>70%).

2.2.4. Berdasarkan pada pola *plagiarisme*

Berdasarkan pada pola *plagiarisme* dibedakan menjadi dua yaitu *plagiarisme* kata demi kata (*word for word plagiarizing*) dan *plagiarisme* mosaik. *Plagiarisme* kata demi kata (*word for word plagiarizing*) adalah pola *plagiarisme* dengan melakukan penjiplakan sebagian kecil kalimat, dapat juga satu paragraf, atau seluruh isi meskipun di tulis dengan bahasa lain. Sedangkan *plagiarisme* mosaik adalah penyajian tidak dilakukan kata demi kata, tapi menyipkan kata atau frase dari beberapa karya orang lain tanpa menuliskan sumbernya sehingga memberi kesan kalimat tersebut adalah asli penulis.

.

2.3. 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*.

2.4. 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. 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.

2.6. 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 .

2.7. *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).

2.8. 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

2.9. 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

2.10. 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 KEMIRIPAN SOURCE CODE*

1. Mengubah isi surce code file acuan dan file pembanding ke dalam kelompok jenis token menggunakan analisis leksikal.
2. Mendeteksi token yang sama di file acuan dan file pembanding.
3. Menghitung tingkat kemiripan source code menggunakan algoritma Smith-Waterman.

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.

1) 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*

2) 3.2.2. Analisis Kebutuhan Perangkat Lunak (Software)

Spesifikasi perangkat lunak yang dibutuhkan antara lain sebagai berikut :

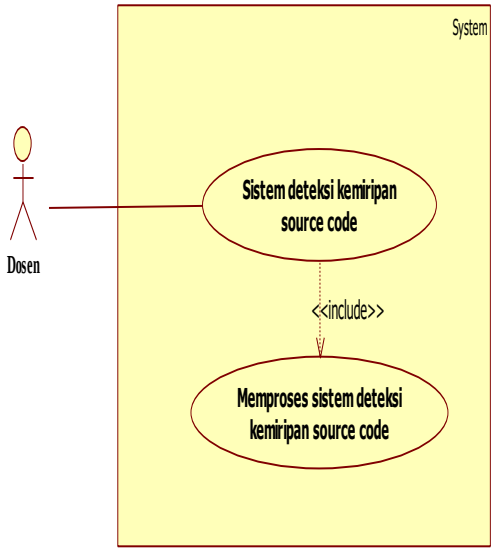
1. Sistem operasi Microsoft *Windows 7 Enterprise 32bit*,
2. Bahasa pemrograman C++

3.3. DESAIN SISTEM

3) 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 :

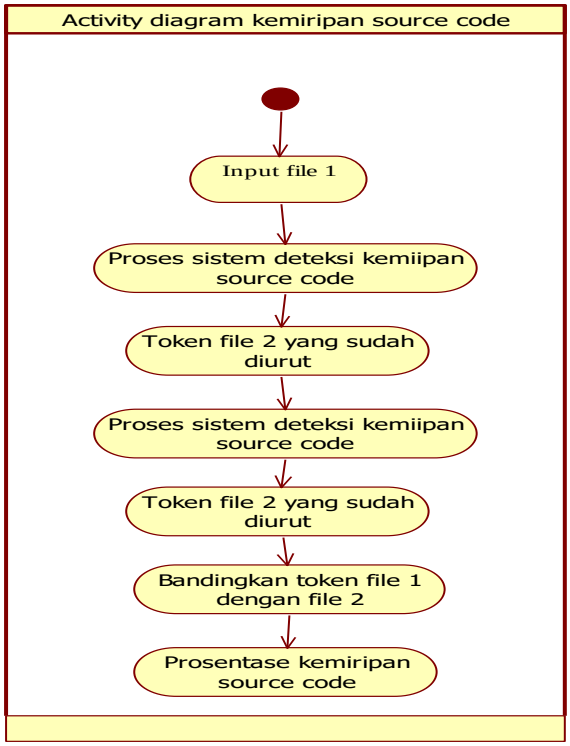


Gambar 3.1. *Use Case Diagram* Sistem

3.3.1.2 Activity Diagram

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

1. Berikut merupakan gambaran *activity* diagram enkripsi algoritma vigenere cipher yang akan dibangun pada gambar 3.2 :



Gambar 3.2. *Activity Diagram* Sistem

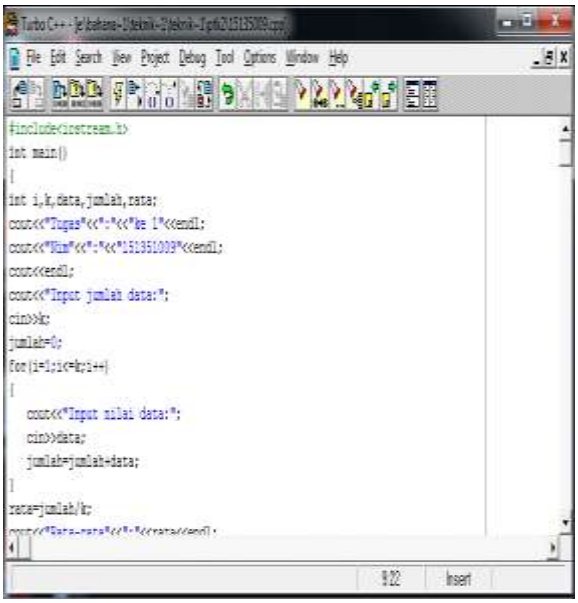
4. Hasil

4. 1. Implementasi

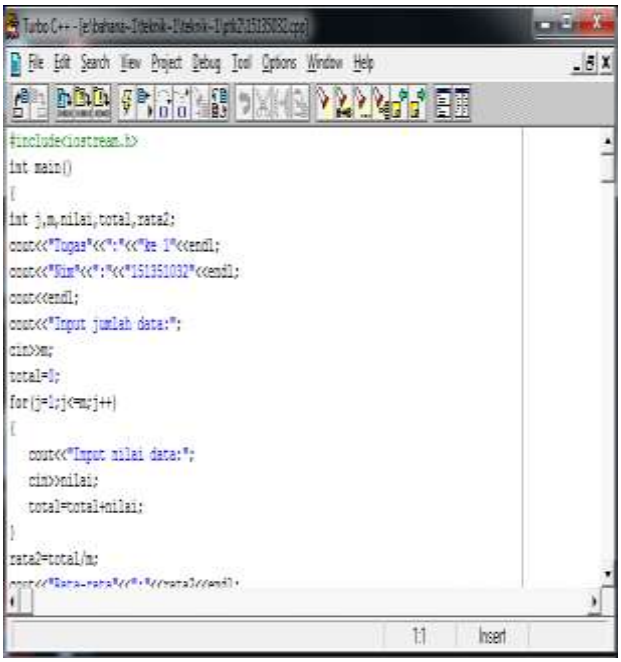
Implementasi merupakan kelanjutan dari kegiatan perancangan sistem. Tahap ini merupakan tahap meletakkan sistem supaya siap untuk di operasikan dan dapat dipandang sebagai usaha untuk mewujudkan sistem yang telah di rancang. Pada tahap ini akan diambil beberapa sampel yang akan diperiksa kemiripan-kemiripan code yang berhasil terdeteksi. Hasil pengujian adalah kemiripan code antara satu code acuan dengan 3 code pembanding yang diambil secara random.

1. File 15135009 dengan file 15135025

Pada sampel pertama, sistem mendeteksi terjadi kemiripan source code di seluruh bagian source code baik milik code acuan maupun pada code pembanding. Pebedaannya disamarkan dengan mengganti isi nim. Isi file source code acuan dan pembanding dapat dilihat di gambar 4.1 dan gambar 4.2., Dengan menggunakan sistem pendeteksi kemiripan source code dapat dideteksi kemiripan source code pada sampel pertama dan diperoleh tingkat kemiripan source code sebesar 94,4% , hasilnya dapat dilihat di gambar 4.3.

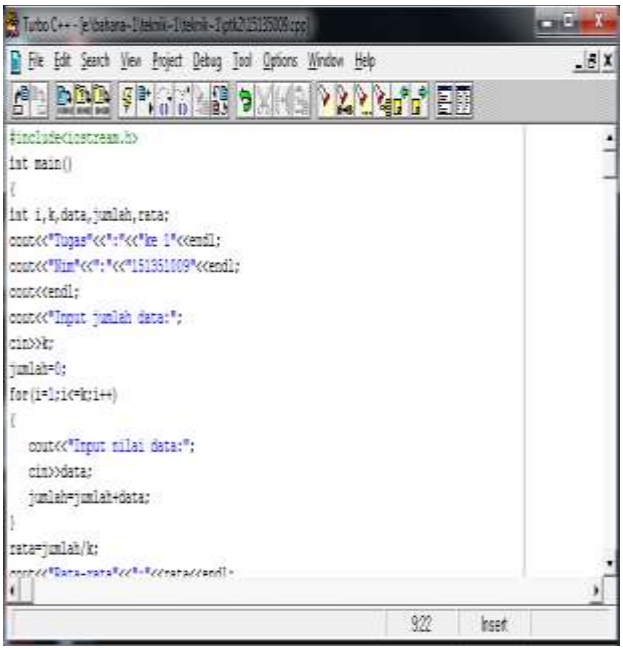


Gambar 4.1. File Acuan Sampel 1



Gambar 4.2. File Pembanding Sampel 1

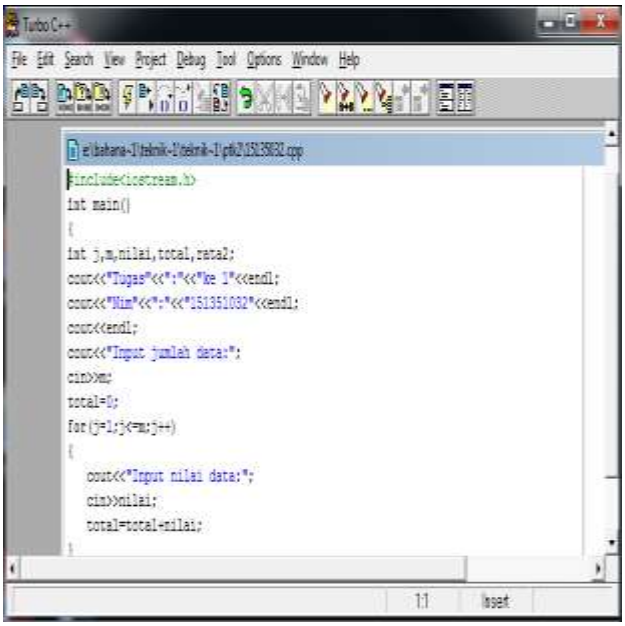
Gambar 4.3 Hasil Deteksi Sampel 1



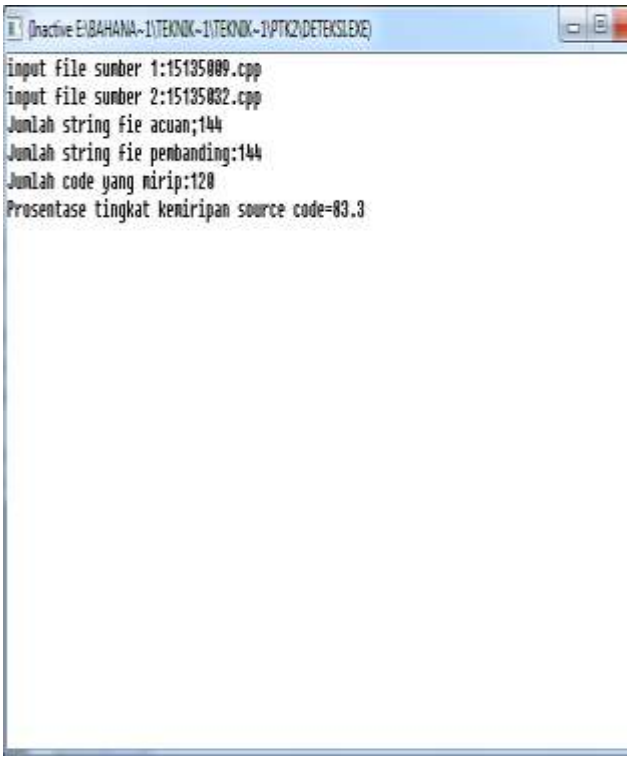
Gambar 4.4. File Acuan Sampel 2

2. File 15135009 dengan file 15135032

Pada sampel kedua, sistem mendeteksi terjadi kemiripan source code di seluruh bagian source code baik milik code acuan maupun pada code pembanding. Pebedaannya disamarkan dengan mengganti semua nama variabel. Isi file source code acuan dan pembanding dapat dilihat di gambar 4.4 dan gambar 4.5., Dengan menggunakan sistem pendeteksi kemiripan source code dapat dideteksi kemiripan source code pada sampel kedua dan diperoleh tingkat kemiripan source code sebesar 83,3%, hasil deteksi kemiripan dapat dilihat di gambar 4.6



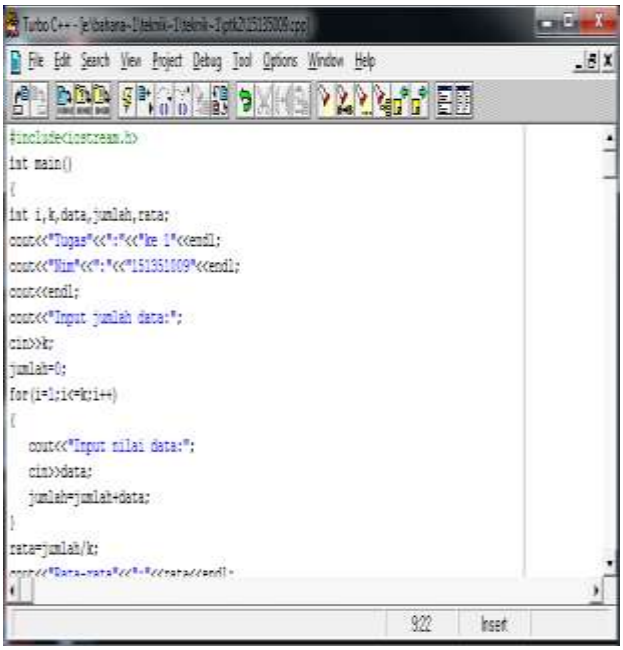
Gambar 4.5. File Pembanding Sampel 2



Gambar 4.6 Hasil Deteksi Sampel 2

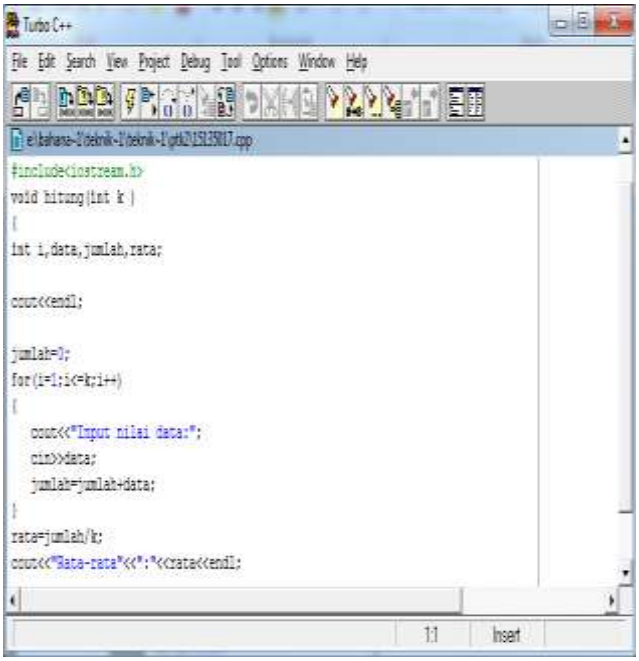
3. File 15135009 dengan file 15135017

Pada sampel ketiga, sistem mendeteksi terjadi kemiripan source code di seluruh bagian source code baik milik code acuan maupun pada code pembanding. Pebedaannya disamarkan dengan memindahkani isi kode program main ke void hitung. Isi file source code acuan dan pembanding dapat dilihat di gambar 4.7 dan gambar 4.8. Dengan menggunakan sistem pendeteksi kemiripan source code dapat dideteksi kemiripan source code pada sampel kedua dan diperoleh tingkat kemiripan source code sebesar 92,7 % , hasil deteksi kemiripan dapat dilihat di gambar 4.9



.

Gambar 4.4. File Acuan Sampel 3



Gambar 4.4. File Pembanding Sampel 3

Gambar 4.9 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	Keterang-an
1	Deteksi kemiripan source code sampel 1	Dapat dideteksi kemiripan source code	Berhasil
2.	Deteksi kemiripan source code sampel 2	Dapat dideteksi kemiripan source code	Berhasil
3.	Deteksi kemiripan source code sampel 3	Dapat dideteksi kemiripan source code	Berhasil

5. KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Setelah melakukan tahap penelitian, perancangan, dan tahap implementasi terhadap pengamanan data akademik dengan menggunakan analisis leksikal, diperoleh kesimpulan sebagai berikut :

1. Analisis leksikal dapat diimplementasikan untuk mendeteksi kemiripan source code dari file source code acuan dan file source code pembanding.
2. Hasil dari aplikasi ini berupa nilai presentase kemiripan yang dapat digunakan sebagai acuan penentuan adanya tindakan plagiarisme.

.

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 kemiripan kode program hanya pada bahasa java, untuk kedepannya diharapkan dapat mendeteksi lebih dari satu bahasa pemrograman.
2. Jumlah file yang diperiksa pada aplikasi ini berjumlah dua buah *source code*, untuk lebih menghemat waktu pemeriksaan kedepannya dapat dikembangkan dengan memeriksaa file lebih dari satu.

DAFTAR PUSTAKA

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

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

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 Hal Europe