BAB II

LANDASAN TEORI

2.1 Algoritma

Dalam matematika dan ilmu komputer, algoritma merupakan langkah-langkah dalam penyelesaian suatu perhitungan. Algoritma umumnya digunakan dalam proses perhitungan, pemrosesan data, dan pengautomasian. Lebih jelasnya, algoritma merupakan metode yang efektif yang ditujukkan pada daftar yang terbatas dari kumpulan perintah yang telah didefinisikan untuk menghitung suatu fungsi. Dimulai dari nilai awal, kemudian kumpulan perintah yang pada saat dieksekusi akan memproses kondisi-kondisi yang telah ditetapkan hingga menghasilkan *output* dan kemudian menentukan kondisi akhir.

Dalam penyelesaian masalah, ada kriteria-kriteria tertentu pada kondisi awal yang harus dipenuhi sebelum mengeksekusi algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria.

Pada sistem komputer, algoritma merupakan gambaran langsung dari logika yang dituliskan oleh pembangun perangkat lunak agar lebih efektif dalam pencapaian target perangkat lunak tersebut, agar dapat memperoleh hasil keluaran dari masukan yang diberikan (terkadang *null*).

Tingkat kerumitan dari suatu algoritma merupakan ukuran banyaknya proses yang dibutuhkan oleh algoritma tersebut dalam menyelesaikan suatu masalah. Dengan kata lain, algoritma yang mampu memecahkan atau menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat

kerumitan yang rendah, sebaliknya algoritma yang membutuhkan waktu yang lama dalam penyelesaian masalah memiliki tingkat kompleksitas yang tinggi.

2.1.1 Jenis-jenis Algoritma

Terdapat beberapa klasifikasi algoritma yang dibagi berdasarkan alasan tersendiri. Salah satu cara dalam melakukan pengklasifikasian tersebut adalah berdasarkan paradigma dan metode yang digunakan dalam perancangan algoritma tersebut. Beberapa paradigma yang digunakan untuk menyusun suatu algoritma antara lain:

1. *Divide and Conquer*, merupakan paradigma untuk membagi suatu permasalahan yang besar menjadi permasalahan-permasalahan yang kecil. Pembagian masalah ini dilakukan secara terus-menerus sampai ditemukan bagian masalah yang kecil dan mudah untuk dipecahkan.
2. *Dynamic programming*, paradigma pemrograman dinamik akan sesuai jika digunakan pada suatu masalah yang mengandung sub-struktur yang optimal dan mengandung beberapa bagian permasalahan yang tumpang tindih. Paradigma ini seklias terlihat mirip dengan paradigma *divide and conquer*, sama-sama mencoba untuk mebagi permasalahan menjadi sub permasalahan yang lebih kecil, tapi secara intrinsic ada perbedaan dari karakter permasalahan yang dihadapi.
3. Metode serakah, merupakan paradigma yang mirip dengan pemrograman dinamik, namun jawaban dari setiap submasalah tidak perlu diketahui dari setiap tahap, dan menggunakan pilihan apa yang terbaik pada saat itu.

4. *Search and enumeration*, merupakan paradigma pemodelan yang memberikan aturan tertentu dalam pemecahan masalah dan optimalisasi.

2.1.2 Analisis Algoritma

Dalam penganalisisan suatu algoritma kita harus memperhatikan beberapa hal, seperti:

1. Kebenaran (*Correctness*)

   Dalam pembuktian kebenaran suatu algoritma, hasil akhir dari algoritma tersebut haruslah diperiksa apakah sudah sesuai dengan kondisi-kondisi yang telah diberikan pada awal masukan. Untuk melakukan pemeriksaan suatu algoritma yang kompleks, kita dapat membagi algoritma tersebut menjadi beberapa modul kecil, sehingga jika modul-kecil tersebut benar maka seluruh program akan benar.

2. Jumlah Operasi yang Dilakukan (*Amount of Work Done*)

   Penghitungan jumlah operasi yang dilakukan ini digunakan untuk membandingkan tingkat efisiensi suatu algoritma dengan algoritma lain dalam memecahkan suatu masalah yang sama. Hal ini dilakukan untuk mendapatkan algoritma yang dapat menghasilkan waktu eksekusi yang lebih cepat. Cara paling mudah dalam membandingkan dua buah algoritma adalah dengan menghitung jumlah operasi dasar yang dilakukan oleh algoritma tersebut, karena apabila dilakukan pembandingan langsung pada komputer, kerap kali kondisi setiap komputer dan cara pembacaan setiap bahasa pemrograman mempengaruhi waktu pemecahan masalah.

3. Analisis Kemungkinan Terburuk (*Worst Case*)

   Analisis *worst case* merupakan analisis yang digunakan untuk melihat tingkat efektifitas suatu algoritma dalam menyelesaikan masalah-masalah yang masukannya merupakan masukan yang terkadang tidak perlu dihitung atau cara mengatasi pada saat kemungkinan masukan salah.

4. Optimal (*Optimality*)

   Untuk menganalisis suatu algoritma, biasanya selalu menggunakan kelas algoritma dan ukuran kompleksitas, misalnya, jumlah operasi dasar yang dilakukan. Sebuah algoritma disebut optimal (untuk *worst case*) jika tidak ada algoritma yang dapat melakukan operasi dasar yang lebih sedikit (untuk *worst case*).

5. Ikatan Terendah (*Lower Bound*)

   Untuk membuktikan bahwa suatu algoritma adalah optimal, tidak diperlukan menganalisis setiap algoritma. Dengan membuktikan teorema-teorema yang menentukan *lower bound* pada jumlah operasi yang diperlukan untuk menyelesaikan masalah, maka algoritma yang dapat melakukan jumlah operasi tersebut disebut optimal.

2.1.3 Mekanisme Pelaksanaan Algoritma oleh Pemroses

Komputer hanyalah salah satu pemroses. Agar dapat dilaksanakan oleh komputer, algoritma harus ditulis dalam notasi bahasa pemrograman sehingga dinamakan program. Jadi program adalah perwujudan atau implementasi teknis algoritma yang ditulis dalam bahasa pemrograman tertentu sehingga dapat dilaksanakan oleh komputer.

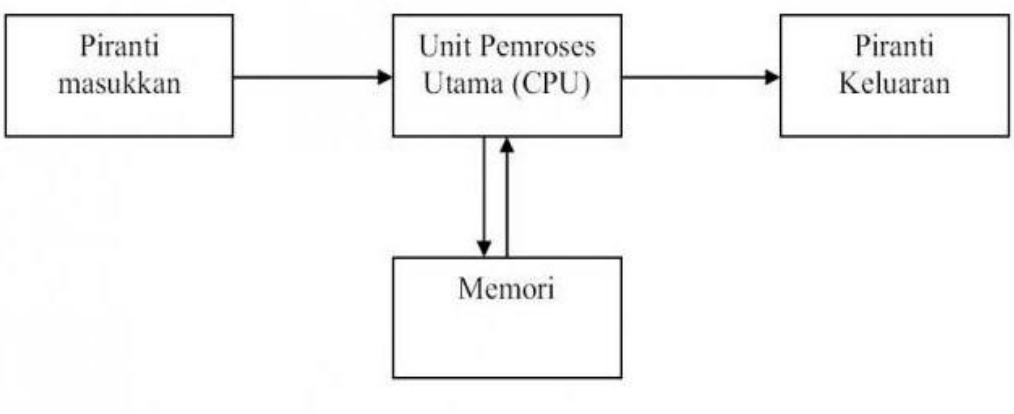
Kata “algoritma” dan “program” seringkali dipertukarkan dalam penggunaannya. Misalnya ada orang yang berkata seperti ini: “program pengurutan data menggunakan algoritma *selection sort*”. Atau pertanyaan seperti ini: “bagaimana algoritma dan program menggambarkan grafik tersebut?”. Jika Anda sudah memahami pengertian algoritma yang sudah disebutkan sebelum ini, Anda dapat membedakan arti kata algoritma dan program. Algoritma adalah langkah-langkah penyelesaikan masalah, sedangkan program adalah realisasi algoritma dalam bahasa pemrograman.

Program ditulis dalam salah satu bahasa pemrograman dan kegiatan membuat program disebut pemrograman (*programming*). Orang yang menulis program disebut pemrogram (*programmer*). Tiap-tiap langkah di dalam program disebut pernyataan atau instruksi**.** Jadi, program tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasi-operasi yang bersesuaian dengan instruksi tersebut dikerjakan komputer.

Secara garis besar komputer tersusun atas empat komponen utama yaitu, piranti masukan, piranti keluaran, unit pemroses utama, dan memori. Unit pemroses utama (*Central Processing Unit – CPU)* adalah “otak” komputer, yang berfungsi mengerjakan operasi-operasi dasar seperti operasi perbandingan, operasi perhitungan, operasi membaca, dan operasi menulis. Memori adalah komponen yang berfungsi menyimpan atau mengingatingat.

Yang disimpan di dalam memori adalah program (berisi operasi-operasi yang akan dikerjakan oleh CPU) dan data atau informasi (sesuatu yang diolah

oleh operasi-operasi). Piranti masukan dan keluaran (*I/O devices)* adalah alat yang memasukkan data atau program ke dalam memori, dan alat yang digunakan komputer untuk mengkomunikasikan hasil-hasil aktivitasnya. Contoh piranti masukan antara lain, papan kunci (*keyboard)*, pemindai (*scanner)*, dan cakram (*disk).* Contoh piranti keluaran adalah, layar peraga (*monitor)*, pencetak (*printer)*, dan cakram.



Gambar 2. 1 Komponen Utama Komputer

(Sumber: http://bebas.vlsm.org/v06/Kuliah/SistemOperasi/BUKU/SistemOperasi-4.X-1/ch01s04.html)

Mekanisme kerja keempat komponen di atas dapat dijelaskan sebagai berikut. Mula-mula program dimasukkan ke dalam memori komputer. Ketika program dilaksanakan (*execute)*, setiap instruksi yang telah tersimpan di dalam memori dikirim ke CPU. CPU mengerjakan operasioperasi yang bersesuaian dengan instruksi tersebut. Bila suatu operasi memerlukan data, data dibaca dari piranti masukan, disimpan di dalam memori lalu dikirim ke CPU untuk operasi

yang memerlukannya tadi. Bila proses menghasilkan keluaran atau informasi, keluaran disimpan ke dalam memori, lalu memori menuliskan keluaran tadi ke piranti keluaran (misalnya dengan menampilkannya di layar monitor).

2.2 Bahasa Pemrograman

Bahasa program merupakan suatu wahana untuk menuangkan pikiran manusia yang dapat dimengerti oleh mesin komputer sehingga bernilai guna. Suatu bahasa program akan terikat aturan dari paradigma bahasa. Ada berbagai macam paradigma bahasa : Prosedural, Fungsional, Deklaratif, *Object Oriented*, Konkuren. Adapun konsep-konsep dasar dalam pemograman adalah sebagai berikut :

1. Simulasi , sensibilitas terhadap masalah dan kemungkinan solusi. Kegiatan dilakukan di kelas, melalui permainan. Contoh : Mengurutkan nama setiap mahasiswa yang ada di kelas dimulai dari huruf A sampai Z. Hal ini dapat dilakukan secara komputerisasi maupun manual.

2. Analisis masalah secara lebih formal dan membuat spesifikasi dan algoritma dalam notasi yang ditetapkan. Mahasiswa harus menuliskan solusi algoritmiknya dalam notasi standar di kelas. Penulisan notasi algoritmik bertujuan untuk menyeragamkan pemahaman tentang algoritma program yang terbebas dari sintak (aturan) penulisan bahasa program

3. Menulis program, yaitu menterjemahkan notasi algoritmik ke dalam sintak bahasa program.

4. *Debugging* dan menguji coba program. Hal ini bertujuan untuk mendapatkan program yang benar. Program dikatakan benar jika terbebas dari salah lojik dan sintak bahasa. Secara ideal mahasiswa hanya diberi kesempatan untuk me-run program sebanyak 2 kali : pertama untuk membersihkan program dari kesalahan sintak dan kedua untuk mendapatkan program benar. Pada tahap ini diharapkan tidak terjadi kesalahan lojik jika analisa benar

5. Mengamati peristiwa eksekusi, perlu dilakukan untuk meningkatkan kepercayaan bahwa jika analisa benar maka sisa pekerjaan menjadi mudah. Pada pemrograman prosedural, aspek ini penting untuk memahami fenomena eksekusi dan perubahaan nilai suatu struktur data.

6. Membaca program : orang akan dapat menulis dengan baik kalau sering membaca. Hal ini juga berlaku dalam memprogram. Kegiatan yang dapat dilakukan di kelas adalah dengan saling tukar menukar teks algoritma, dan saling mengkritik algoritma teman. Mahasiswa harus berlatih sendiri pada kegiatan belajar bersama.

7. Membuktikan kebenaran program secara formal , satu-satunya hal yang menjamin kebenaran, tetapi kontradiktif dan sulit diterapkan dalam kehidupan sehari-hari. Program yang hanya lima baris pembuktiannya bisa sehalaman, sehingga seringkali tidak pernah diterapkan dalam aplikasi nyata. Aktifitas ini dicakup dalam matakuliah Analisis Algoritma.

2.3 Beda Algoritma dan Program

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman. Beberapa pakar memberi formula bahwa :

Program = Algoritma + Bahasa (Struktur Data)

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

Pembuatan algoritma mempunyai banyak keuntungan di antaranya :

1. Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
2. Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
3. Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

Beberapa hal yang perlu diperhatikan dalam membuat algoritma :

1. Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
2. Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
3. Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
4. Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya.
5. Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.
6. Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam

notasi bahasa pemrograman. Ada beberapa hal yang harus diperhatikan pada translasi tersebut, yaitu :

a. Pendeklarasian variabel

   Untuk mengetahui dibutuhkannya pendeklarasian variabel dalam penggunaan bahasa pemrograman apabila tidak semua bahasa pemrograman membutuhkannya.

b. Pemilihan tipe data

   Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu hal ini dipertimbangkan pada saat pemilihan tipe data.

c. Pemakaian instruksi-instruksi

   Beberapa instruksi mempunyai kegunaan yang sama tetapi masing-masing memiliki kelebihan dan kekurangan yang berbeda.

d. Aturan sintaksis

   Pada saat menuliskan program kita terikat dengan aturan sintaksis dalam bahasa pemrograman yang akan digunakan.

e. Tampilan hasil

Pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini diperhatikan ketika mengkonversikannya menjadi program.

f. Cara pengoperasian compiler atau interpreter.

Bahasa pemrograman yang digunakan termasuk dalam kelompok compiler atau interpreter.

2.4 Penjadwalan Kuliah

Penjadwalan kuliah merupakan proses pembuatan jadwal kegiatan kuliah yang selalu mengawali pergantian semester pada setiap perguruan tinggi. Proses pembuatan jadwal kuliah ini haruslah memperhatikan berbagai aspek yang mempengaruhi jadwal kuliah, seperti banyaknya matakuliah, ketersediaan ruangan serta rentang waktu yang dibutuhkan untuk setiap matakuliah. Proses inti pada penjadwalan kuliah adalah proses untuk memasangkan komponen-komponen tertentu seperti matakuliah, dosen, ruangan, dan waktu dengan dipengaruhi sejumlah batasan serta persyaratan tertentu. Permasalahan yang sering muncul dalam proses penjadwalan kuliah oleh para penjadwal adalah terdapatnya jauh lebih banyak jumlah matakuliah dan kelas yang harus dijadwalkan dengan jumlah ruangan, persyaratan akan matakuliah tertentu untuk mendapat kelas tertentu (matakuliah tertentu terkadang membutuhkan fasilitas seperti laboratorium), serta sering kali ada keinginan para pengajar untuk mengajar pada suatu hari atau jam tertentu.

Dalam pengaturan jadwal kuliah ini, para staff Tata Usaha (Sekretariat Jurusan) harus melalui beberapa tahapan, seperti melakukan pendataan jumlah matakuliah dan sksnya, pendataan kurikulum, pendataan pengajar, kelas-kelas yang disediakan, serta syarat dan prioritas-prioritas yang dipakai. Setelah itu staff Tata Usaha harus mengatur dan melakukan pemasangan untuk setiap matakuliah, dosen, ruangan, kelas, dan waktu ke dalam jadwal kuliah. Kemudian tahap terakhir adalah memeriksa kembali dan mengevaluasi prioritas-prioritas dan syarat-syarat yang ada, apakah semua sudah dipenuhi atau belum.

Karena itu kerap kali proses penjadwalan kuliah untuk suatu perguruan tinggi memakan waktu yang sangat lama sehingga mahasiswa kesulitan dalam menentukan pengambilan kelas untuk mata kuliah tertentu (untuk hal di mana mahasiswa dapat memilih kelas, misalkan mahasiswa konversi dan mahasiswa yang ingin mengambil matakuliah atas dengan sks lebih).

Untuk mengatasi permasalahan yang ada, terutama masalah waktu yang dibutuhkan, sejumlah algoritma telah dikembangkan untuk menyelesaikan dan menyusun data penjadwalan kuliah ini, salah satunya adalah algoritma yang digunakan penulis dalam karya tulis ini, yaitu Algoritma Genetika.

2.5 Perangkat Lunak Pendukung Perancangan Aplikasi

Dalam pembuatan program aplikasi, untuk laporan tugas akhir ini penulis menggunakan beberapa perangkat lunak yang menunjang pembuatan program aplikasi berbasis desktop, yaitu windows sebagai sistem operasi yang akan dipakai, java sebagai bahasa pemrograman, mysql sebagai database, sqlYog

sebagai editor database, serta netbeans sebagai platform framework dan IDE pembangunan aplikasi.

2.5.1 Java

Java merupakan bahasa pemrograman yang awalnya dikembangkan oleh James Gosling di Sun Microsystem. James Gosling, Mike Sheridan, dan Patrick Naughton memulai proyek untuk bahasa pemrograman Java awalnya adalah untuk industry televise interaktif, namun bahasa Oak (nama awal dari Java) terlalu maju untuk teknologi televisi. Bahasa pemrograman ini menggunakan bahasa gabungan yang mirip dengan bahasa C dan C++.

Sun Microsystem merilis implementasi public pertama untuk Java 1.0 pada tahun 1991. Prinsip perilisan ini adalah "*Write Once, Run Anywhere*". Bahasa pemrograman ini menyediakan tingkat keamanan yang tinggi dan menyediakan fiture untuk mengkonfigurasi keamaanannya. Segera setelah perilisannya, banyak web browser yang memasukkan kemampuan untuk menjalankan applet dari Java yang membuat bahasa pemrograman ini segera populer. Dengan munculnya Java 2 (dirilis pada awalnya sebagai J2SE pada Desember 1998-1999), versi ini dibangun dengan memiliki konfigurasi diberbagai platform. J2EE ditargetkan untuk menangani aplikasi-aplikasi enterprise, J2ME ditargetkan untuk menangani aplikasi mobile, dan J2SE ditargetkan untuk aplikasi-aplikasi standard. Pada tahun 2006 untuk alas an pemasaran, Sun mengganti nama J2 untuk masing-masing edisi menjadi Java EE, Java ME, dan Java SE.

Ada lima target utama dari bahasa Java dalam pembangunan untuk setiap aplikasi, yakni:

1. Simpel, berbasis objek dan umum
2. Kuat dan aman
3. Memiliki arsitektur yang netral dan portable
4. Dieksekusi dengan performa kinerja yang tinggi
5. Harus bias ditafsirkan, terulir, dan dinamis

2.5.2 Microsoft Office Access

Microsoft Access (atau Microsoft Office Access) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi Microsoft Office, selain tentunya Microsoft Word, Microsoft Excel, dan Microsoft PowerPoint. Aplikasi ini menggunakan mesin basis data Microsoft Jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna.

Microsoft Access dapat menggunakan data yang disimpan di dalam format Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/programmer yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. Access juga mendukung teknik-teknik

pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

2.5.3 Netbeans

NetBeans awalnya dibangun pada tahun 1996 sebagai Xelfi (untuk pemrograman Delphi) oleh seorang mahasiswa dari Charles University di Paraguai. Pada tahun 1997, Roman Stanek membangun sebuah perusahaan dan merilis versi komersial dari NetBeans hingga akhirnya dibeli oleh Sun MicroSystem pada 1999. Hingga saat ini platform NetBeans telah banyak berkembang di bawah SunMicrosystem.

Netbeans merupakan platform framework dan IDE (*integrated development environtment*) yang digunakan untuk pengembangan aplikasi desktop yang menggunakan bahasa Java, dan beberapa bahasa lain, seperti Groovy, C, C++ dan banyak lagi.

Netbeans IDE dibangun menggunakan bahasa Java dan dapat dijalankan pada Windows, OS X, Linux, Solaris dan sistem operasi lain yang mendukung JVM. IDE NetBeans merupakan alat pengembangan aplikasi yang terintegrasi. NetBeans IDE mendukung pengembangan program yang menggunakan bahasa Java dari semua versi (Java SE, Java ME, Java EE).

Platform NetBeans memperbolehkan pembangunan aplikasi dengan menggunakan modul-modul. Aplikasi yang dibangun menggunakan netBeans dapat dikembangkan oleh pihak ketiga. Platform NetBeans merupakan platform

yang dapat digunakan ulang (*reusable*) untuk mempermudah pembangunan program menggunakan bahasa Java.

Platform menyediakan layanan yang *reusable* pada aplikasi desktop, yang mempermudah pembangun untuk fokus pada spesifikasi logik dari aplikasi. Fungsi yang disediakan dari platform neh antara lain:

1. *User Interface Management* (menu dan toolbars)
2. *User Setting Management* (menangani pengaturan)
3. *Storage Management* (menyimpan dan membuka berbagai jenis data)
4. *Window Management*
5. *Wizard Framework* (mendukung tahapan berdasarkan langkah-langkah)
6. *NetBeans Visual Library*
7. *Integrated development tools*