ALAT BANTU AJAR ALGORITMA PENGURUTAN

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika

Oleh:

Nama : **Rini Ida Lestari**
No. Mahasiswa : **99 523 158**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2004

LEMBAR PENGESAHAN

ALAT BANTU AJAR ALGORITMA PENGURUTAN

TUGAS AKHIR

Oleh :

Nama : **Rini Ida Lestari**
No. Mahasiswa : 99 523 158

Yogyakarta, April 2004
Pembimbing

(Fathul Wahid, ST, MSc)

LEMBAR PENGESAHAN PENGUJI

ALAT BANTU AJAR ALGORITMA PENGURUTAN

TUGAS AKHIR

Oleh :

Nama : Rini Ida Lestari
No. Mahasiswa : 99 523 158

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk
Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 27 April 2004

Tim Penguji :

Fathul Wahid, ST, MSc
Ketua

Sri Kusumadewi, SSi, MT
Anggota I

Wawan Indarto, ST
Anggota II

Mengetahui
Dekan Fakultas Teknologi Industri
Universitas Islam Indonesia

Ir. H. Bachrun Sutrisno, MSc

HALAMAN PERSEMBAHAN

Alhamdulillah Hirobil 'Alamin

Penulis mempersembahkan Tugas Akhir ini kepada :

Ayahanda, Noordin, SH

Ibunda, Sri Lestariani (alm)

Adik-adikku, Hendra dan Lilis

dan Muhammad Fauzan, ST

Atas semua doa, dorongan dan dukungannya.....

HALAMAN MOTTO

"*Sesungguhnya ramah-tamah (lemah-lembut) di dalam segala urusan akan menjadikan urusan itu indah (sukses). Tanpa sikap lemah lembut, pastilah semua urusan akan menjadi buruk.*" (H.R Muslim)

"*Orang bijak adalah orang yang mengamalkan ilmunya kepada orang lain, sehingga kala dia tiada maka namanya akan tetap terkenang.*"

KATA PENGANTAR

Assalamu 'alaikum Wr. Wb.

Segala puji bagi Allah SWT, yang senantiasa melimpahkan rahmat dan hidayah-Nya kepada kita semua. Alhamdulillah, dengan perkenan-Nya penulis dapat menyelesaikan Laporan Tugas Akhir ini yang berjudul Alat Bantu Ajar Algoritma Pengurutan.

Laporan Tugas Akhir ini dibuat sebagai syarat untuk memperoleh gelar kesarjanaan, selain itu juga berfungsi sebagai sarana menuangkan gagasan dan ide di masa akhir penulis menuntut ilmu di bangku perkuliahan.

Semoga dengan tulisan yang sederhana ini dapat memperkaya khazanah penelitian yang sudah menjadi tradisi mahasiswa sebagai pemuda intelektual bagi bangsa dan agamanya.

Selesainya tugas akhir ini tentunya tidak terlepas dari segenap pihak yang membantu dan mendukung secara moril maupun materiil, oleh karena itu penulis ucapkan terima kasih kepada segenap pihak yang telah membantu terselesaikannya tugas akhir ini :

1. Bapak Ir. H. Bachrun Sutrisno, MSc, selaku dekan FTI UII.
2. Ibu Sri Kusumadewi, SSi, MT, selaku ketua jurusan Teknik Informatika.
3. Bapak Fathul Wahid, ST. MSc, selaku pembimbing tugas akhir.
4. Ayahanda, Ibunda, dan adik-adikku tercinta atas doa, kepercayaan, dukungan, dan kasih sayangnya.
5. Seluruh keluarga besarku terimakasih banyak untuk doa dan dukungannya.
6. Mas Fauzan thank you so much untuk semua kesabaran, dukungan, doa dan kasih sayangnya selama ini.
7. Mas Ardi thanx a lot atas kesabarannya dalam memberi pelajaran algoritma.
8. Untuk sahabat terbaikku Achie, keceriaan dan kebersamaan kita selama ini sangat berkesan semoga itu akan terus berlanjut, buat Yudi jagain sahabatku!
9. Untuk sahabatku Ellin buruan nyusul dan tetep semangat, dan untuk teman-temenku yang gila abiz Rifqi & Haris tetep berjuang ya.

10. Rekan-rekan Teknik Informatika angkatan 1999 pada khususnya dan semua mahasiswa Teknik Informatika UII pada umumnya.
11. Semua pihak yang tidak dapat penulis sebutkan satu persatu, terima kasih atas dorongan dan motivasinya.

Tugas akhir ini tentunya tidaklah lepas dari kesalahan dan kekurangan, oleh karena itu penulis mengharapkan saran dan kritik dari pembaca.

Billahitaufiq wal Hidayah,

Wassalamu 'alaikum Wr. Wb.

Yogyakarta, 26 April 2004

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

ABSTRAKSI

Tugas akhir ini berjudul Alat Bantu Ajar Algoritma Pengurutan yang dibuat untuk membantu para siswa dalam mempelajari algoritma pengurutan. Perangkat lunak yang dibuat merupakan aplikasi alat bantu ajar dengan menggunakan simulasi kartu dan juga dengan proses yang bertahap. Dengan demikian siswa dapat melihat bagaimana suatu metode algoritma pengurutan dieksekusi baris per baris.

Dengan dibuatnya Alat Bantu Ajar Algoritma Pengurutan ini diharapkan para siswa dapat lebih menyukai dan lebih mudah memahami proses pengurutan. Karena perangkat lunak ini dibuat secara visual, maka siswa tidak perlu lagi membayangkan suatu proses pengurutan. Semua akan terlihat jelas pada simulasi kartu, oleh karena itu algoritma pengurutan akan terlihat lebih mudah dan menarik dibandingkan bila hanya mempelajari algoritma dalam bentuk teks.

Prosedur-prosedur pengurutan dituangkan dalam bahasa pemrograman Pascal dan dibangun dengan software Borland Delphi 6 yang menunjang bahasa Pascal. Bahasa pemrograman Pascal digunakan karena relatif lebih mudah dipelajari dan juga merupakan bahasa pemrograman terstruktur.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada umumnya lembaga-lembaga pendidikan khususnya bidang komputer, memberikan pelajaran algoritma secara konvensional kepada siswanya, yaitu pengajar memberikan materi pelajaran di depan kelas atau dengan menggunakan bantuan papan tulis tanpa adanya praktek dan simulasi. Hal tersebut dikarenakan terbatasnya sarana belajar untuk para siswa, yang dalam hal ini adalah laboratorium. Sehingga hal tersebut sedikit menghambat proses belajar siswa dalam menyerap dan memahami apa yang diberikan oleh para pengajarnya, karena tidak adanya sarana yang dapat mensimulasikan algoritma-algoritma yang diberikan oleh para pengajar.

Berdasarkan keadaan tersebut, pada saat ini hampir semua lembaga pendidikan khususnya bidang komputer berlomba-lomba melengkapi dirinya dengan memiliki suatu laboratorium. Salah satunya adalah laboratorium algoritma dan pemrograman yang berfungsi untuk mempermudah proses mengajar dimana didalamnya dilengkapi alat bantu ajar yang dapat mensimulasikan algoritma-algoritma yang diajarkan.

1.2 Rumusan Masalah

Untuk mempermudah proses belajar-mengajar algoritma, maka rumusan masalah yang muncul dalam penelitian ini adalah : Bagaimana membuat alat bantu ajar mengenai algoritma pengurutan yang dilengkapi dengan simulasi.

1.3 Batasan Masalah

Mengingat kompleknya permasalahan dalam penelitian ini, maka lingkup permasalahan dalam penelitian perlu dibatasi pada hal-hal sebagai berikut :

1. Alat bantu ajar ini hanya membahas metode algoritma pengurutan berikut ini :
   - Metode *count sort*.
   - Metode *bubble sort*.
   - Metode *selection sort*.
   - Metode *Insertion Sort*.
2. Data (kartu) yang akan diurutkan sebanyak 10 kartu.
3. Pada saat proses pengurutan berlangsung, program tidak dapat menerima interupsi apapun.
4. Bahasa pemrograman yang digunakan adalah bahasa Pascal.
5. Program pembangun *interface* menggunakan Borland Delphi 6.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Merancang suatu sistem yang dapat mempermudah siswa dalam mempelajari dan memahami algoritma-algoritma yang diberikan oleh para pengajar..
2. Membuat perangkat lunak yang dapat mensimulasikan algoritma-algoritma didalamnya.

1.5 Manfaat Penelitian

Penelitian ini bermanfaat untuk merancang dan merealisasikan perangkat lunak alat bantu ajar yang dapat mensimulasikan algoritma-algoritma didalamnya, sehingga dapat mempermudah siswa dalam memahami algoritma yang diberikan.

1.6 Metodologi Penelitian

Keberhasilan suatu alat bantu belajar tergantung pada metode pengambilan keputusan yang tepat, metode itu harus mencakup seluruh faktor-faktor yang berkaitan dengan masalah yang akan dipecahkan. Berikut ini adalah penjelasan lebih lanjut tentang perancangan perangkat lunak (*software*) untuk membantu mempelajari algoritma pengurutan :

1. Investigasi masalah

   Pada tahap ini, yang dilakukan adalah menentukan dan mengamati masalah yang ada sehingga dapat dilakukan analisis mengenai metode yang akan diterapkan pada masalah.

2. Analisis kebutuhan

   Pada tahap ini dilakukan analisis kebutuhan perangkat lunak, bagaimana perangkat lunak dapat dibangun dan dapat mengetahui kelemahannya.

3. Perancangan perangkat lunak

   Pada tahap ini membahas mengenai bagaimana perangkat lunak dibentuk, direncanakan dan pembuatan sketsa.

4. Implementasi perangkat lunak.

   Implementasi perangkat lunak yang telah dirancang ke dalam bahasa pemrograman.

5. Analisis kinerja

   Tahap ini berupa analisis kinerja terhadap perangkat lunak yang telah diimplementasikan ke dalam bahasa pemrograman.

1.6.1 Metodologi Pengumpulan Data

Dalam penelitian ini dibutuhkan data-data yang relevan dan dapat digunakan untuk memecahkan permasalahan yang diteliti.

1. Pengumpulan data sekunder

Pengumpulan data yang berasal dari sumber selain dari objek penelitian, yang dalam hal ini dimaksudkan untuk menggali teori yang akan mendukung terhadap penelitian dalam memecahkan masalah. Data ini diperoleh dari laporan atau referensi yang berhubungan dengan penelitian.

2. Penelitian kepustakaan

Penelitian ini dilakukan dalam rangka pengumpulan beberapa data untuk referensi, dengan mendalami buku-buku serta literatur-literatur yang berhubungan dengan masalah yang dihadapi.

1.7 Sistematika Penulisan

Dalam penulisan laporan dilakukan dengan cara sebagai berikut :

Bab II berisi tentang teori-teori yang menjadi acuan untuk pelaksanaan penelitian yang meliputi teori-teori tentang algoritma pengurutan.

Bab III berisi tentang segala sesuatu yang dibutuhkan oleh penulis untuk membuat tugas akhir ini, menjelaskan metode analisis yang memuat uraian tentang metode analisis kebutuhan perangkat lunak yang dipakai.

Bab IV membahas mengenai metode perancangan dan hasil perancangan, yang meliputi struktur data, arsitektur perangkat lunak, rincian prosedur-prosedur, dan antamuka yang dikembangkan.

Bab V memuat dokumentasi implementasi perangkat lunak dan keterangan tentang implementasi dari rancangan perangkat lunak.

Bab VI yaitu analisis kinerja perangkat lunak, memuat dokumentasi hasil pengujian terhadap perangkat lunak.

Bab VII berisi kesimpulan dari proses perancangan perangkat lunak serta saran-saran.

BAB II

LANDASAN TEORI

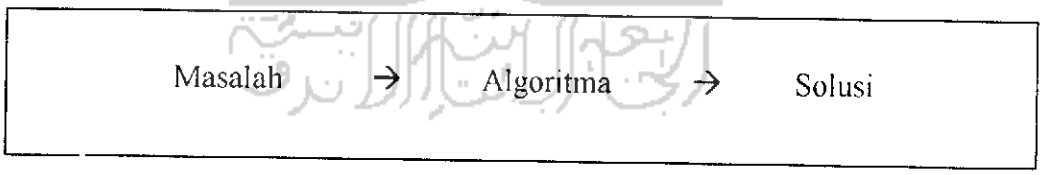
2.1 Definisi Algoritma

Algoritma berasal dari kata *algoris* dan *ritmis*. Istilah ini pertama kali dungkapkan oleh Abu Ja'far Mohammed Ibn Musa al Khowarizmi (825 M) dalam buku *Al-Jabr Wa-al Muqobla* [SUT00].

"*Algoritma adalah urutan langkah berhingga untuk memecahkan masalah logika atau matematika* " [PRA00].

Sedangkan dalam bidang pemrograman, algoritma didefinisikan sebagai "*suatu metode khusus yang tepat dan terdiri dari serangkaian langkah yang terstruktur dan dituliskan secara sistematis, yang akan dikerjakan untuk menyelesaikan suatu masalah dengan bantuan komputer* " [SUT00].

Hubungan antara algoritma, masalah dan solusi ditinjukkan pada gambar 2.1 :



Gambar 2.1 Diagram hubungan masalah, algoritma dan solusi

Proses dari masalah hingga menjadi suatu algoritma disebut tahap pemecahan masalah, sedangkan tahap dari algoritma hingga menjadi suatu solusi

disebut dengan tahap implementasi. Solusi yang dimaksud adalah suatu program yang merupakan implementasi dari algoritma disusun.

Dalam kehidupan sehari-hari, sebenarnya algoritma digunakan untuk melakukan sesuatu. Sebagai contoh, jika ingin menulis surat, maka perlu dilakukan langkah berikut :

1. Mempersiapkan kertas dan amplop.
2. Mempersiapkan alat tulis, seperti pena atau pensil.
3. Mulai menulis.
4. Memasukkan kertas ke dalam amplop.
5. Pergi ke kantor pos untuk mengeposkan surat tersebut.

Langkah-langkah dari nomor 1 sampai dengan nomor 5 di atas itulah yang disebut dengan algoritma. Jadi sebenarnya algoritma digunakan baik sadar maupun tidak sadar.

2.2 Karakteristik Algoritma

1. Algoritma harus tidak rancu (*unambiguous*)

   Deskripsi langlah-langkah dalam algoritma harus hanya mempunyai tafsiran tunggal. Karenanya, sebuah bahasa pemrograman seperti pascal sering dignakan untuk menuliskan algoritma, sehingga langkah-langka yang ditliskan hanya mempunyai tafsiran tunggal, terutama menurut komputer.

2. Algoritma harus tepat (*precise*)

   Algoritma harus menyatakan urutan langkah-langkahnya. Kapan sebuah aksi atau langkah x dilakukan, apakah sebelum aksi y atau sesudahnya, harus

dinyatakan dengan jelas. Algoritma harus juga menyatakan dengan jelas kapan berhenti dari sebuah aksi untuk meneruskan ke aksi berikutnya.

3. Algoritma harus pasti (*definite*)

   Jika serangkaian aksi yang sama dilakukan dua kali maka hasilnya harus selalu sama. Sebagai ilustrasi, jika dua orang koki mengikuti resep yang sama maka rasa kue yang dihasilkan harus sama. Kalau hasil akhir atau rasa kue tidak sama, pasti ada beberapa perbedaan yang tidak disadari, seperti merek bahan yang berbeda, atau lama memasak yang berbeda, atau ada perbedaan pada suatu yang lain.

4. Algoritma harus berhingga (*finite*)

   Serangkaian langkah dalam algoritma harus dapat dilaksanakan pada rentan waktu tertentu seperti yang telah diuraikan di atas [WAH00].

2.3 Algoritma dan Pemrograman Terstruktur

Konsep pemrograman terstruktur memegang peranan penting dalam merancang, menyusun, memelihara dan mengembangkan suatu program, khususnya program aplikasi yang besar dan kompleks.

Sebelum mempelajari pemrograman terstruktur lebih lanjut, ada beberapa istilah mendasar yang perlu dipahami lebih dahulu, yaitu :

a. Program adalah kata, ekspresi, pernyataan atau kombinasinya yang disusun atau dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman sehingga dapat dieksekusi oleh komputer.

b. Bahasa pemrograman merupakan prosedur/tata cara penulisan program. Pada bahasa pemrograman terdapat dua faktor penting, yaitu sintaks dan semantik. Sintaks adalah urutan-urutan gramatikal yang mengatur tata cara penulisan kata, ekspresi dan pernyataan, sedangkan semantik adalah aturan-aturan untuk menyatakan suatu arti.

c. Pemrograman merupakan proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengan menggunakan suatu bahasa pemrograman.

d. Pemrograman terstruktur merupakan proses mengimplementasikan urutan langkah untuk menyelesaikan masalah dalam bentuk program yang memiliki rancang bangun yang terstruktur dan tidak berbelit-belit sehingga mudah ditelusuri, dipahami dan dikembangkan oleh siapa saja [SUT00].

2.3.1 Ciri Teknik Pemrograman Terstruktur

Teknik pemrograman terstruktur memiliki ciri atau karakteristik sebagai berikut :

1. Mengandung algoritma pemecahan masalah yang tepat, benar, sederhana, standar dan efektif.
2. Memiliki struktur logika dan struktur program yang benar dan mudah dipahami.
3. Membutuhkan biaya testing, pemeliharaan dan pengembangan yang rendah.
4. Memiliki dokumentasi yang baik [SUT00].

2.3.2 Langkah-langkah dalam Pemrograman Komputer

Dalam melakukan suatu kegiatan, tentu saja diperlukan langkah-langkah yang harus dilalui. Dalam pemrograman komputer, juga memerlukan beberapa langkah. Berikut ini adalah beberapa langkah yang harus dilakukan dalam pemrograman komputer :

1. Mendefinisikan masalah
2. Analisis kebutuhan
3. Penyusunan algoritma
4. Pengkodean/pemrograman
5. Testing dan debuging
6. Pemeliharaan
7. Dokumentasi

2.4 Pengurutan

Pengurutan (*sorting*) secara umum didefinisikan sebagai proses pengaturan kembali serangkaian obyek dalam rangkaian tertentu [WAH00]. Tujuan proses pengurutan adalah untuk memudahkan proses pencarian seperti pada buku telepon, kamus, daftar isi buku atau majalah, berkas-berkas dan lain-lain yang sering dijumpai di dalam perpustakaan, di dalam gudang, bahkan hampir di setiap tempat penyimpanan obyek yang suatu saat akan dicari kembali. Jika suatu data tidak terurut, maka pencarian suatu data akan membutuhkan waktu yang sangat panjang dan memerlukan tingkat ketelitian yang sangat tinggi. Karena itu pengurutan merupakan bagian yang sangat penting dan relevan khususnya dalam pemrosesan data.

Ada dua macam urutan yang biasa digunakan dalam proses pengurutan, yaitu urut naik atau membesar dan urut turun atau mengecil. Sebagai contoh data bilangan bulat 5, 2, 6, dan 4 dapat diurutkan naik menjadi 2, 4, 5, 6, atau diurutkan turun menjadi 6, 5, 4, 2 [PRA00].

Pengurutan data juga merupakan contoh yang bagus untuk menunjukkan bahwa suatu persoalan dapat diselesaikan dengan sejumlah algoritma yang berbeda-beda, dimana masing-masing mempunyai keuntungan dan kerugian yang harus diperhitungkan untuk suatu aplikasi tertentu [WIR97].

Dengan demikian pengurutan merupakan suatu aktivitas yang penting khususnya dalam pengolahan data. Meskipun demikian ketertarikan utama pada pengurutan adalah hal teknik dasar yang digunakan untuk membentuk sejumlah algoritma. Secara khusus pengurutan merupakan subyek ideal untuk menunjukkan adanya sejumlah variasi algoritma yang semuanya bertujuan sama. Beberapa diantaranya terkesan cukup efisien dan mempunyai keuntungan lebih dibanding yang lain. Dengan demikian merupakan suatu hal yang ideal untuk mensimulasikan proses-proses algoritma pengurutan ke dalam suatu bentuk yang mudah dipahami.

2.5 Metode Pengurutan Data

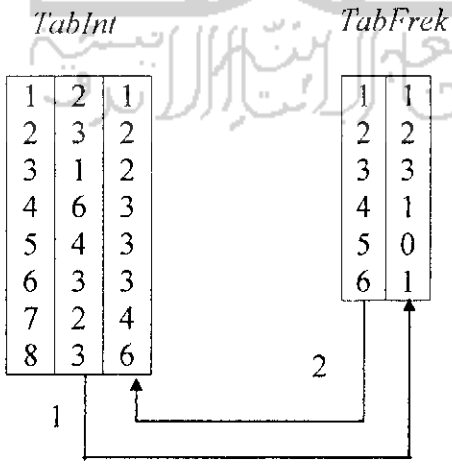
Ada beberapa metode pengurutan data yang dikenal, antara lain : metode pencacahan, metode pemilihan, metode penukaran, metode penyisipan.

2.5.1 Pengurutan dengan Pencacahan

Pengurutan dengan pencacahan disebut juga *sorting by counting* atau *count sort*. Ide dasar pengurutan dengan pencacahan adalah menghitung frekuensi kemunculan setiap elemen array yang digunakan untuk mengisikan kembali elemen array dari yang terkecil sampai yang terbesar. Setiap elemen diisikan sebanyak frekuensi kemunculannya [WAH00].

Untuk melakukan proses pengurutan dibutuhkan sebuah array untuk mencatat frekuensi kemunculan setiap elemen array. Sebagai contoh dengan nama TabFrek[*min*...*max*] dimana *min* adalah nilai terkecil elemen array dan *max* adalah nilai terbesarnya. TabFrek[*min*..*max*] diinisialisasi dengan nol yang kemudian diisi dengan frekuensi kemunculan elemen pada data TabInt[1..*n*]. setelah frekuensi kemunculan diisikan dalam array TabFrek[*min*..*max*], TabInt[1..*n*] diisi ulang dengan menuliskan indeks TabFrek mulai dari nilai *min* sampai *max* masing-masing sebanyak TabFrek[*i*], dimana *i* = *min*..*max*.

Gambar 2.3 adalah contoh metode pengurutan dan pencacahan :



Gambar 2.2 Pengurutan dengan pencacahan

Dari gambar 2.3 di atas dapat dijabarkan dengan algoritma sebagai berikut :

1. Asumsi nilai min dan max diketahui TabInt[1..n].
2. Dari i ← min sampai max lakukan langkah 3.
3. TabFrek[i] ← 0.
4. Dari i ← 1 sampai n lakukan langkah 5.
5. TabFrek[TabInt[i]] ← TabFrek[TabInt[i]] + 1.
6. j ← 0.
7. Dari i ← min sampai max lakukan langkah 8.
8. Jika TabFrek[i] <> 0 maka kerjakan langkah 9.
9. k ← 1.
10. j ← j+1.
11. TabInt[j] ← i.
12. k ← k+1.
13. Ulangi langkah 10, 11,12 sampai k>TabFrek[i].

2.5.2 Pengurutan dengan Pemilihan

Pengurutan dengan pemilihan atau seleksi disebut juga *sorting by selection* atau *selection sort.* Ide dasar pengurutan dengan pemilihan adalah mencari nilai ekstrim dengan array dan menukarnya dengan elemen yang paling ujung yang tidak diikutkan dalam proses selanjutnya [WAH00]. Kemudian hal yang sama dilakukan pada array sisanya. Pemilihan nilai ekstrim tersebut tergantung pada hasil akhir pengurutan yang diinginkan. Jika hasil akhir

pengurutan yang diinginkar array terurut membesar maka nilai ekstrim yang dipilih adalah nilai terkecil, dan sebaliknya.

Adapun proses pengurutan dengan pemilihan adalah mula-mula dilakukan pengulangan dari 1 sampai dengan $(N-1)$. Pada tiap-tiap pengulangan dicari data terkecil dari data ke $(i+1)$ sampai dengan data terakhir $(=N)$. Data terkecil ini kemudian ditukarkan dengan Pivot, yaitu data ke-i. tentu saja apabila data terkecil tersebut lebih besar daripada data ke-i maka proses penukaran tidak perlu dilakukan [PRA00]. Proses ini dapat diuraikan pada algoritma sebagai berikut :

1. $i \leftarrow 1$.
2. Selama $i <= N-1$ kerjakan baris 3 sampai dengan 9.
3. $k \leftarrow i$.
4. $j \leftarrow i+1$.
5. Selama $(j <= N)$ kerjakan baris 6 dan 7.
6. Jika (Data [k] > Data [j]) maka $k \leftarrow j$.
7. $j \leftarrow j+1$.
8. Tukar Data [i] dengan Data [k].
9. $i \leftarrow i+1$.

Dari algoritma di atas dapat disimpulkan bahwa jumlah perbandingan $(-C)$ adalah sebagai berikut :

$$C = \frac{N(N-1)}{2} = \frac{(N^2-N)}{2} \quad \text{.......................................(2.2)}$$

Adapun jumlah penukaran (=M) yang dilakukan tergantung pada keadaan datanya dimana jumlah penukaran minimum, maksimum, dan rata-rata adalah sebagai berikut :

$$M_{min} = (3N-1) \quad \text{.....(2.3)}$$

$$M_{max} = \frac{N^2}{4} + 3(N-1) \quad \text{.....(2.4)}$$

$$M_{rata-rata} = N(LogN + 0{,}57) \quad \text{.....(2.5)}$$

2.5.3 Pengurutan dengan Penukaran

Pengurutan dengan penukaran disebut juga *sorting by exchange* atau *bubble sort.* Ide dasar pengurutan dengan penukaran adalah dengan mengapungkan elemen yang nilainya kecil dengan pertukaran, sehingga akan terurut membesar dengan asumsi seperti gelembung [WAH00]. Pemeriksaan elemen dilakukan dengan membandingkan dua elemen yang berdekatan mulai dari indeks terbesar (n) sampai indeks terkecil (1). Tahapan ini akan menghasilkan nilai terkecil pada indeks terkecil. Selanjutnya dilakukan proses yang sama tanpa melibatkan elemen pada indeks yang telah terurut.

Adapun algoritma untuk pengurutan dengan penukaran adalah sbagai berikut :

1. $i \leftarrow 2$.
2. Selama ($i <= N-1$) kerjakan baris 3 sampai dengan 7.
3. $j \leftarrow N$.
4. Selama ($j >= i$) kerjakan baris 5 sampai dengan 7.

5. Jika (Data [j-1] > Data [j]) maka tukar Data [j-1] dengan Data [j].

6. $j \leftarrow j-1$.

7. $i \leftarrow i+1$.

Dari algoritma di atas dapat disimpulkan bahwa jumlah perbandingan (= C) selalu tetap, yaitu :

$$C = \frac{N(N-1)}{2} = \frac{(N^2 - N)}{2} \quad \ldots\ldots(2.6)$$

Adapun jumlah penukaran (=M) yang dilakukan tergantung pada keadaan datanya dimana jumlah penukaran minimum, penukaran rata-rata dan penukaran maksimum adalah sebagai berikut :

$$M_{\min} = 0 \quad \ldots\ldots(2.7)$$

$$M_{\max} = \frac{3(N^2 - N)}{2} = (N^2 - N) \times 1{,}5 \quad \ldots\ldots(2.8)$$

$$M_{rata-rata} = \frac{3(N^2 - N)}{4} = (N^2 - N) \times 0{,}75 \quad \ldots\ldots(2.9)$$

Jumlah penukaran minimum terjadi jika data sudah dalam keadaan urut dan jumlah penukaran maksimum terjadi bila data dalam keadaan urut terbalik [PRA00].

2.5.4 Pengurutan dengan Penyisipan

Pengurutan dengan penyisipan disebut juga *sorting by insertion* atau *insertion sort*. Ide dasar pengurutan dengan penyisipan adalah dengan melakukan proses sekuensial terhadap array dan untuk menemukan tempat yang sesuai untuk setiap elemen sehingga semua elemen array terurut. Ide ini dapat dianalogkan dengan pengurutan kartu pada tangan [WAH00].

Untuk n elemen, elemen pertama (indeks = 1) dianggap sudah terurut. Kemudian elemen pada indeks selanjutnya (indeks = i) dicarikan tempat sesuai diantara indeks 1 sampai dengan indeks i. Proses ini dilakukan sampai semua elemen sudah menempati tempat yang tepat (indeks = n).

Adapun algoritma untuk pengurutan dengan penyisipan adalah sebagai berikut :

1. $i \leftarrow 2$.
2. Selama ($i <= N$) kerjakan baris 3 sampai dengan 10.
3. $x \leftarrow Data[i]$.
4. Data[0] $\leftarrow x$
5. $j \leftarrow i-1$.
6. Selama ($x < Data[j]$) kerjakan baris 7 dan 8.
7. Data $[j+1] \leftarrow Data[j]$.
8. $j \leftarrow j-1$.
9. Data $[j+1] \leftarrow x$.
10. $i \leftarrow i+1$.

Dari algoritma di atas dapat disimpulkan bahwa jumlah perbandingan (=C) dan pergeseran (=M) tergantung pada keadaan data dengan perhitungan jumlah perbandingan minimum, perbandingan rata-rata, dan perbandingan maksimum adalah sebagai berikut :

$$C_{min} = 2(N-1) \quad \ldots\ldots(2.10)$$

$$C_{rata-rata} = \frac{N^2+N+2}{4} \quad \ldots\ldots(2.11)$$

$$C_{max} = \frac{N^2 + N - 2}{2} \quad \text{.....(2.12)}$$

Jumlah perbandingan minimum terjadi jika data sudah dalam keadaan urut dan jumlah perbandingan maksimum terjadi bila data dalam keadaan urut terbalik [PRA00].

Adapun perhitungan jumlah pergeseran minimum, pergeseran rata-rata, dan pergeseran maksimum sebagai berikut :

$$M_{min} = 2(N-1) \quad \text{.....(2.13)}$$

$$M_{rata-rata} = \frac{N^2 - 7N - 8}{4} \quad \text{.....(2.14)}$$

$$M_{max} = \frac{N^2 + 3N - 4}{2} \quad \text{.....(2.15)}$$

2.6 Bahasa Pemrograman Borland Delphi

Borland Delphi adalah bahasa pemrograman yang bekerja dalam lingkup Microsoft Windows yang dapat memanfaatkan kemampuan Microsoft Windows secara optimal. Kemampuan Borland Delphi secara umum adalah menyediakan komponen-komponen yang memungkinkan seorang programmer membuat program aplikasi yang sesuai dengan tampilan dan cara kerja Microsoft Windows. Selain itu, Borland Delphi diperkuat dengan bahasa pemrograman terstruktur yang sangat handal, yaitu struktur bahasa pemrograman Object Pascal yang sangat terkenal [ALA00].

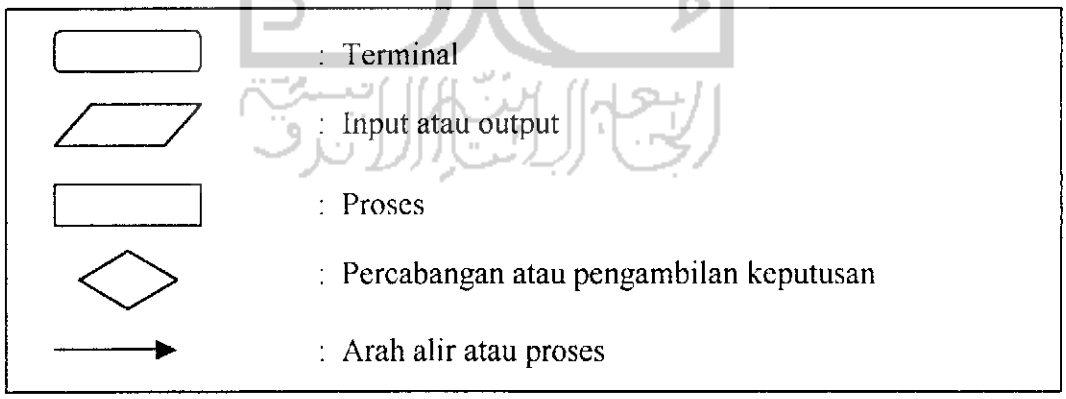
Borland Delphi memiliki beberapa tipe data dengan karakter dan ukuran bytes yang berbeda-beda, adapun untuk penggunaan data random pada Borland Delphi dengan cara menentukan banyaknya data dahulu baru diisi.

Contoh :

```
Randomize ;
jum_data :=strtoint (ed_jumlah.Text);
Randomize;
sg_acak.RowCount:=jum_data+1;
for i:=1 to jum_data do
      begin
          data:=RandomRange(0,9999);
          arrInt [i]:=data;
          sg_acak.Cells[0,i]:=inttostr(i);
           sg_acak.Cells[1,i]:=inttostr(data);
       end;
```

2.7 Teknik Pemrograman

Berdasarkan teori yang ada, dalam membuat suatu perangkat lunak diperlukan teknik pemrograman sebagai dasar pembuatan program. Teknik tersebut dapat berupa Data Flow Diagram (DFD), Relasional Database, atau flowchart. Adapun untuk membuat perangkat lunak mengenai metode pengurutan data diperlukan Flowchart dengan simbol-simbol yang ditunjukkan oleh gambar 2.4 :



Gambar 2.3 Simbol-simbol Flowchart

BAB III

ANALISIS KEBUTUHAN PERANGKAT LUNAK

3.1 Analisis Sistem

Analisis sistem (*system analysis*) dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan.

Analisis sistem berfungsi sebagai jembatan antara pengalokasian fungsi pada perangkat lunak. Hal tersebut memungkinkan didefinisikannya fungsi-fungsi dan kinerja perangkat lunak, antarmuka perangkat lunak dengan elemen sistem lainnya, dan kekangan-kekangan yang harus dipenuhi oleh perangkat lunak.

3.2 Metode Analisis

Metode analisis yang digunakan untuk menganalisis kebutuhan perangkat lunak ini adalah metode analisis terstruktur. Metode analisis terstruktur merupakan teknik pemodelan isi dan aliran informasi. Dalam hal ini terdapat input, proses dan output yang dinyatakan dengan diagram alir (*flowchart*).

Setelah hal diatas dilakukan, penyempurnaan dilaksanakan secara bertahap dengan tetap menjaga konsistensi aliran informasi dan tanpa menginjak pada aspek prosedural.

3.3 Hasil Analisis

Dari metode analisis di atas diperoleh hasil analisis berdasarkan landasan teori yang ada dimana terdapat perbandingan efektifitas masing-masing metode pengurutan data yang akan diimplementasikan ke dalam pembuatan sistem. Berikut ini adalah beberapa hasil analisis yang telah diperoleh :

3.3.1 Analisis Masalah

Beragamnya metode pengurutan data menjadikan perlunya visualisasi yang berupa simulasi untuk dapat membedakan metode pengurutan yang satu dengan yang lainnya, dan memperjelas langkah-langkah pengurutan itu sendiri.

3.3.2 Masukan Sistem

Input atau masukan berupa indeks random kartu yang akan diurutkan dan metode pengurutan data yang akan dipakai.

3.3.3 Keluaran Sistem

Output atau keluaran berupa data yang telah diurutkan dengan visualisasi berupa simulasi menggunakan kartu.

3.3.4 Kebutuhan Fungsi

Untuk mengimplementasikan algoritma pengurutan ke dalam aplikasi alat bantu ajar algoritma pengurutan yang akan dibuat diperlukan fungsi-fungsi sebagai berikut :

1. Fungsi pengacakan (*randomize*) data.

2. Fungsi pengurutan data dengan metode-metode pengurutan data yang ada di batasan masalah.
3. Fungsi visualisasi pengurutan dengan simulasi kartu.
4. Fungsi pemilihan langkah-langkah algoritma pengurutan yang akan disimulasikan.

3.3.5 Analisis kebutuhan perangkat keras

Perangkat keras minimal yang digunakan agar aplikasi ini dapat berjalan dengan baik adalah 1 unit personal komputer dengan spesifikasi sebagai berikut :

1. *Processor* Pentium I 133 MHz atau yang sekelasnya.
2. Ram (*Random Acces Memory*) 32 Mb.
3. VGA *card* atau AGP *card* 2 MB.
4. *Free memory harddisk* 100 MB.
5. Monitor VGA atau SVGA.
6. *Mouse*.
7. *Keyboard*.

3.3.6 Sistem Operasi

Sistem operasi yang dibutuhkan untuk mengimplementasikan dan mengembangkan aplikasi ini adalah sistem operasi Microsoft Win9x atau di atasnya.

BAB IV

PERANCANGAN PERANGKAT LUNAK

4.1 Metode Perancangan

Dalam merancang perangkat lunak implementasi algoritma metode-metode pengurutan data pada beberapa bahasa pemrograman, digunakan pengembangan dari metode analisis yang telah dilakukan. Dari data-data yang diperoleh akan dilakukan perancangan suatu perangkat lunak yang sekiranya akan menjadi sumber informasi mengenai perbandingan lamanya waktu pemrosesan untuk beberapa metode pengurutan data yang ada.

Adapun metode perancangan yang akan dilakukan yaitu pembuatan flowchart terlebih dahulu kemudian pembuatan rancangan antar muka (*user interface*) sebagai pedoman pembuatan form pada bahasa pemrograman.

4.2 Hasil Perancangan

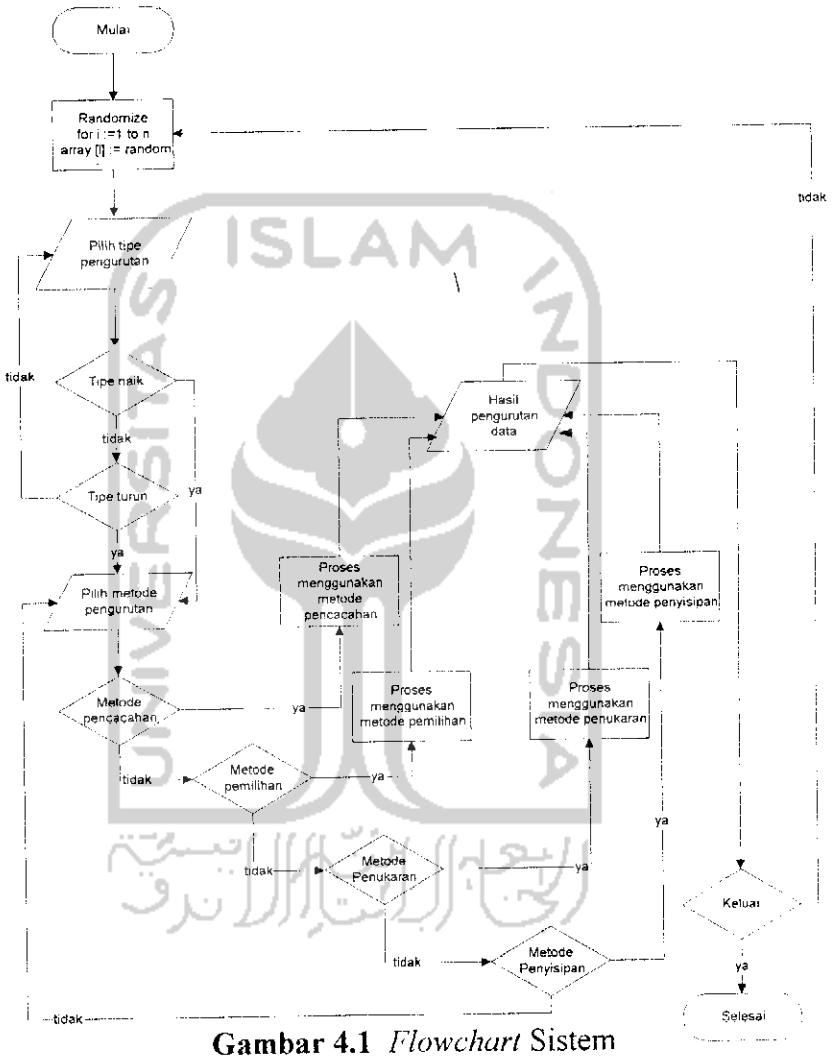
Sebagaimana telah disebutkan pada metode perancangan, berikut ini adalah flowchart dan rancangan antar muka sebagai hasil perancangan perangkat lunak. Keseluruhan hasil perancangan ini akan diimplementasikan dalam perangkat lunak metode-metode pengurutan data pada beberapa bahasa pemrograman berdasarkan prosedur algoritma pengurutan data.

4.2.1 Perancangan Diagram Alir Sistem (*Flowchart*)

Metode diagram alir sistem dengan alat bantu *flowchart* digunakan untuk mendeskripsikan sistem perangkat lunak yang akan dibangun secara mendetail,

sehingga akan memudahkan dalam memahami proses sistem sekaligus dalam pembuatan sistem perangkat lunak tersebut.

Flowchart sistem dapat dilihat pada gambar 4.1 :



Gambar 4.1 *Flowchart* Sistem

Adapun *pseudocode* masing-masing metode pengurutan data adalah sebagai berikut :

1. Metode Pencacahan

Pada pengurutan metode pencacahan ini dibutuhkan dua buah array, yaitu TabInt untuk menyimpan data acak dan data setelah diurutkan dan TabFrek untuk menyimpan frekuensi/jumlah kemunculan data. Dari nilai terkecil sampai nilai terbesar dicari jumlah kemunculan data yang disimpan pada array TabInt. Kemudian data tersebut diisikan kembali ke array TabInt berdasarkan masing-masing kemunculannya. *Pseudocode* pengurutan dengan metode *count sort* adalah sebagai berikut :

```
Procedure CountSort;
var
  TabFrek   : array [min..max] of integer;
  n,i,j,k   : integer;
begin
  {asumsi : TabInt [1..n] terisi;
  n, max, dan min diketahui}
  {Inisialisasi TabFrek}
  for i:=min to max do
      TabFrek[i]:=0;
  {menghitung frekuensi}
  for i:=1 to n do
    TabFrek[TabInt[i]]:=TabFrek[TabInt[i]]+1;
  j:=0;
  for i:=min to max do
  if TabFrek[i]<>0 then
    begin
      k:=1;
      repeat
        j:=j+1;
        TabInt [j]:=i;
        k:=k+1;
      until (k>TabFrek[i])
    end;
end;
```

2. Metode Pemilihan

Pada pengurutan metode pemilihan ini pencarian nilai minimum dilakukan pada array TabInt kemudian menukarkannya dengan nilai ujung kiri untuk pengurutan membesar ke atas. Proses tersebut diulang sampai banyaknya data, tetapi nilai terujung (sudah terurut) tidak diikutkan lagi. *Pseudocode* pengurutan dengan metode *selection sort* adalah sebagai berikut :

```
Procedure SelectionSort;
var
    TabInt          :array [min..max] of integer;
    n, i, j         :integer;
    max, temp       :integer;
begin
    {asumsi : TabInt[1..n] terisi; n, max, dan min diketahui}
    for i:=1 to n-1 do
      max:=i;
    for j:=i+1 to n do
      if TabInt [j]<TabInt[max] then
          max:=j;
          temp:=TabInt[max];
          TabInt[max]:=TabInt[i];
          TabInt[i]:=temp;
end;
```

3. Metode Penukaran

Pada pengurutan metode penukaran ini nilai terkecil ditempatkan pada awal array (kiri) sedangkan data yang lain menggeser ke kanan mengisi kekosongan tanpa me.ubah formasinya. *Pseudocode* pengurutan dengan metode *bubble sort* adalah sebagai berikut :

```
Procedure BubbleSort;
var
  i,j         :integer;
  temp        :integer;
begin
  {asumsi:TabInt[1..n] terisi; n diketahui}
  for i:=2 to n do
  begin
      for j:=n down to i do
      if TabInt[j]<TabInt[j-1] then
      begin         {penggeseran}
        temp:=TabInt[j-1];
```

```
        TabInt[j-1]:=TabInt[j];
        TabInt[j]:=temp;
      end;
  end;
end;
```

4. Metode Penyisipan

Pada pengurutan metode penyisipan ini pengurutan dilakukan dengan mencari tempat yang sesuai untuk nilai setiap elemen array sehingga semua elemen array terurut. *Pseudocode* pengurutan dengan metode *Insertion sort* adalah sebagai berikut :

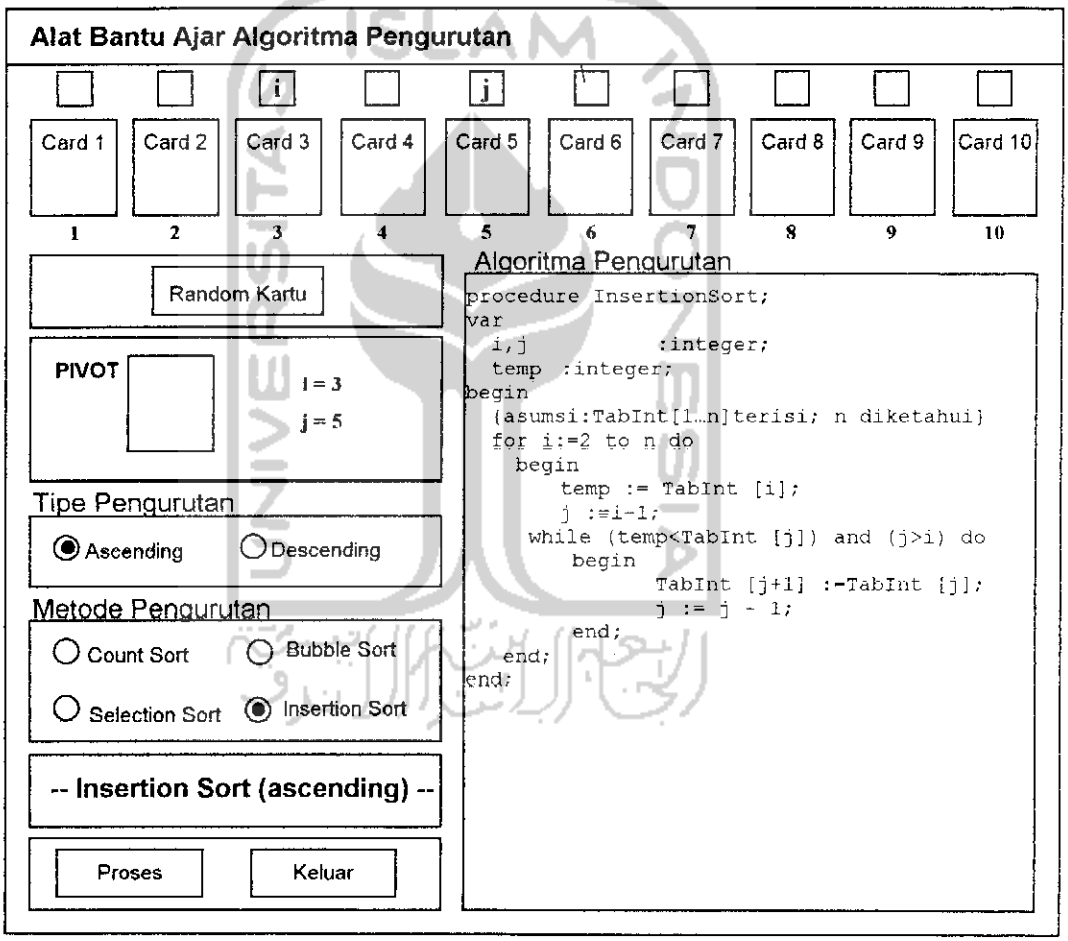
```
Procedure InsertionSort;
var
  i,j       :integer;
  temp      :integer;
begin
  {asumsi:TabInt[1…n]terisi; n diketahui}
  for i:=2 to n do
    begin
      temp := TabInt [i];
      j :=i-1;
      while (temp<TabInt [j]) and (j>i) do
        begin
            TabInt [j+1] :=TabInt [j];
            j := j - 1;
      end;          {temp>=TabInt or j<=1}
      if temp>=TabInt [-] then
           TabInt [j + 1] :=temp
      else          {temp<TabInt[j]}
         begin
           TabInt [j + 1] :=TabInt [j];
           TabInt [j] := temp
         end;
    end;
end;
```

4.2.2 Rancangan Antar Muka

Rancangan antar muka untuk aplikasi alat bantu ajar pengurutan data adalah dengan perancangan model visual.

4.2.2.1 Rancangan Antar Muka Proses Utama

Pada *form* ini user akan memilih metode pengurutan yang diinginkan dengan meng-klik *radio group* kemudian dapat memulai proses pengurutan dengan menekan tombol proses. Simulasi akan terlihat pada *image* kartu di bagian atas *form* setelah tombol proses ditekan. Untuk lebih jelasnya mengenai rancangan antarmuka sistem pada gambar 4.2.



Gambar 4.2 Rancangan Antar Muka Sistem

BAB V

IMPLEMENTASI PERANGKAT LUNAK

5.1 Implementasi Secara Umum

Implementasi perangkat lunak Alat Bantu Ajar Algoritma Pengurutan, menggunakan bahasa pemrograman visual yaitu Borland Delphi 6. Tahap implementai ini merupakan akumulasi dari seluruh analisa sistem yang dilakukan di awal. Hal-hal yang berkaitan dengan fungsi dan prosedur juga kebutuhan akan pengembangan sistem dimanifestasikan ke dalam sistem nyata.

Proses implementasi akan menyesuaikan dari keseluruhan analisa yang telah dilakukan. Hasil analisa kemudian diterjemahkan ke dalam bahasa pemrograman yang didekomposisikan menjadi fungsi-fungsi dan prosedur secara terstruktur.

5.2 Alasan Pemilihan Bahasa Pemrograman

Beberapa alasan penggunaan bahasa pemrograman visual Borland Delphi 6 antara lain adalah :

1. Bahasa pemrograman Borland Delphi 6 merupakan bahasa visual yang dapat digunakan dengan mudah dalam membangun suatu sistem perangkat lunak. Program ini berjalan pada sistem operasi *Microsoft Windows*.
2. Borland Delphi 6 merupakan pengembangan dari bahasa pemrograman *Pascal* yang memiliki sistem pemrograman prosedural.
3. Tersedianya banyak komponen-komponen visual yang cukup refresentatif untuk digunakan dalam membangun aplikasi berbasis *Windows*, dan

kemampuannya untuk menambah komponen-komponen visual baru sesuai kebutuhan.

5.3 Batasan Implementasi

Aplikasi yang dibangun adalah Alat Bantu Ajar Algoritma Pengurutan. Aplikasi ini hanya memberikan gambaran mengenai proses-proses pengurutan yang divisualisasikan dengan kartu-angka. Selain itu aplikasi ini juga memberikan tampilan prosedur-prosedur dan langkah-langkah yang sedang dijalankan pada suatu proses pengurutan.

5.4 Tahapan Pembuatan Perangkat Lunak

Pada pembuatan aplikasi Alat Bantu Ajar Algoritma Pengurutan dengan bahasa pemrograman Borland Delphi 6 dibangun dengan tahapan sebagai berikut :

1. Tahapan perumusan algoritma pengurutan

   Tahapan ini dilakukan untuk mengetahui algoritma pengurutan yang akan digunakan pada masing-masing algoritma pengurutan.

2. Tahapan pemrograman visual

   Pada tahap ini, yang dilakukan adalah membuat *form* hasil rancangan analisis serta kontrol-kontrol program pada setiap *form* tersebut, dengan menggunakan komponen-komponen yang telah tersedia pada Borland Delphi 6.

3. Tahap penulisan kode

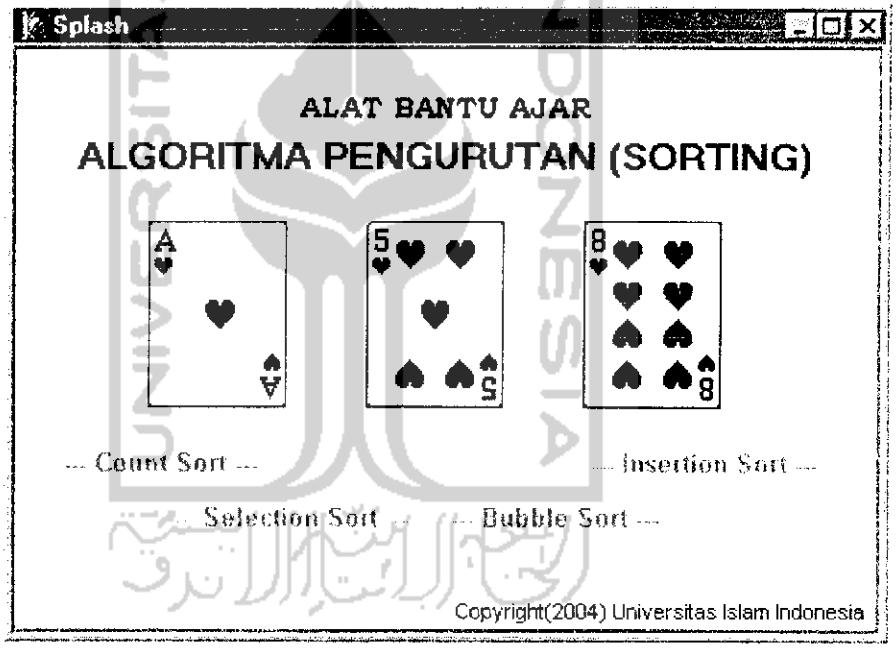
   Tahapan penulisan kode adalah mendeskripsikan kompone-komponen objek yang digunakan pada *form*, kemudian disusun dalam bentuk pengkodean untuk menggabungkan antar komponen yang telah dideskripsikan. Penulisan

kode dilakukan dengan menggunakan editor kode yang tersedia pada Borland Delphi 6.

5.5 Implementasi Antar Muka

5.5.1 *Form* Pembuka (*Splash Form*)

Form pembuka adalah antar muka awal ketika aplikasi diaktifkan, *form* ini akan muncul beberapa detik kemudian akan hilang secara otomatis disusul dengan munculnya *form* utama (*form* pengurutan). Tampilan dari *form* pembuka dapat dilihat pada gambar 5.1 :

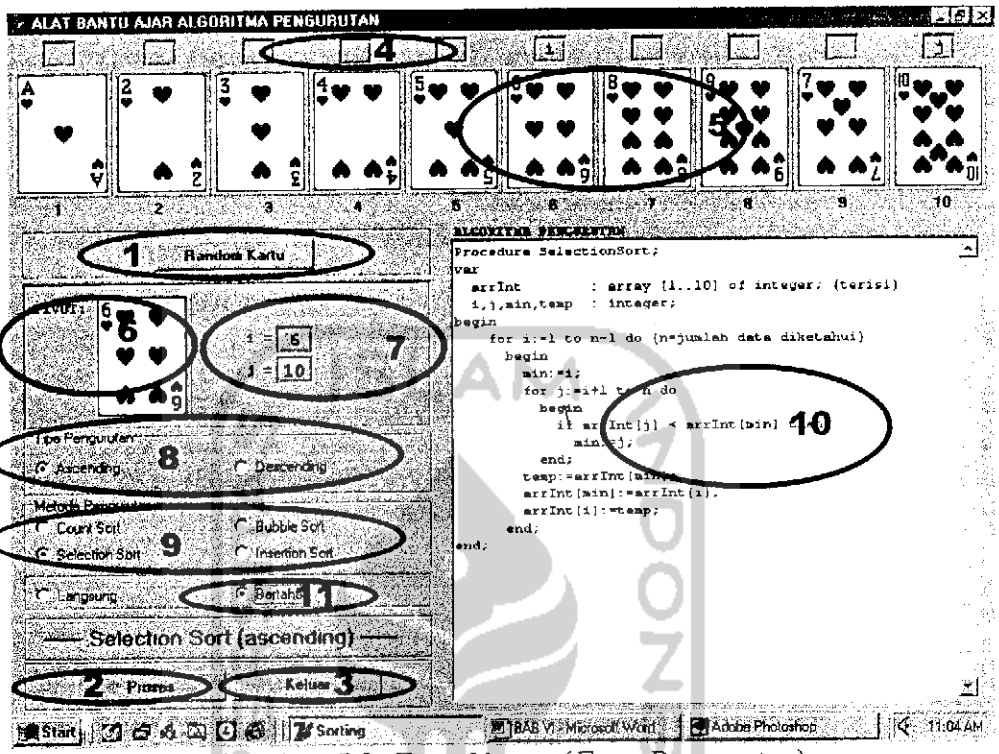


Gambar 5.1 *Form* Pembuka (*Splash Form*)

5.5.2 *Form* Utama (*Form* Pengurutan)

Form utama (*form* pengurutan) adalah *form* dimana semua algoritma pengurutan diimplementasikan dan divisualisasikan. *Form* ini dilengkapi dengan

simulasi kartu dan kode editor yang menunjukkan algoritma pemrograman yang sedang dilakukan. Tampilan dari *form* utama dapat dilihat pada gambar 5.2 :



Gambar 5.2 *Form* Utama (*Form* Pengurutan)

Keterangan dari komponen-komponen yang ada pada *form* utama adalah sebagai berikut :

1. Tombol Random Kartu

   Tombol ini berfungsi untuk mengacak kartu sehingga akan muncul pada panel gambar kartu yang telah teracak.

2. Tombol Proses

   Berfungsi untuk melakukan pengurutan serta sekaligus mensimulasikan pada kartu dan pada kode editor.

3. Tombol Keluar

   Berfungsi untuk keluar dari aplikasi ini.

4. Panel Variabel

   Berfungsi untuk menunjukkan posisi variabel-variabel yang digunakan (dalam hal ini variabel i dan j).

5. Panel Gambar Kartu

   Berfungsi untuk menampilkan dan mensimulasikan gambar kartu yang akan diurutkan.

6. Panel Gambar Pivot

   Menampilkan gambar pivot yang ada pada proses pengurutan.

7. Panel Nilai i dan j

   Menampilkan nilai dari variabel i dan variabel j pada saat program dieksekusi.

8. *Radio Group* Tipe Pengurutan

   Untuk Memilih tipe pengurutan yang akan digunakan yaitu *ascending* dan *descending.*

9. *Radio Group* Metode Pengurutan

   Untuk Memilih metode pengurutan yang akan digunakan; berisi *Count Sort, Selection Sort, Bubble Sort* dan *Insertion Sort.*

10. Kode Editor Algoritma Pengurutan

    Berfungsi hanya menampilkan algoritma pengurutan saja, dan tidak dapat di-*edit.* Saat proses pengurutan dilakukan, kode editor akan menampilkan *highlight* pada baris-baris program yang sedang dieksekusi.

11. *Radio Group* Proses Pengurutan

    Berfungsi untuk memilih proses langsung atau bertahap.

5.6 Prosedur dan Algoritma

5.6.1 Teknik Pemrograman

Teknik pemrograman yang digunakan adalah teknik prosedural, artinya sistem secara keseluruhan didekomposisi menjadi prosedur yang saling berkaitan. Alat pemrograman yang digunakan adalah Borland Delphi 6 yang sudah cukup mudah digunakan, karena masih mendukung teknik pemrograman prosedural.

Dengan menggunakan Borland Delphi 6 secara otomatis *file* yang dibuat akan terbentuk menjadi empat jenis *file*, yaitu *file* dengan ekstensi *dpr, pas, dfm,* dan *dcu.* Jika program dikompilasi maka akan dihasilkan *file* dengan ekstensi *exe.*

5.6.2 Prosedur-prosedur dalam Program

Pemrograman pada aplikasi Alat Bantu Ajar Algoritma Pengurutan terdiri dari prosedur-prosedur yang dapat memudahkan dalam pembuatan program. Berikut ini akan ditampilkan beberapa bagian prosedur besera pembahasannya.

5.6.2.1 Prosedur Random

Prosedur random kartu berfungsi untuk mengacak nilai dari array yang akan diurutkan dalam proses pengurutan.

```
procedure TfSorting.bRandomClick(Sender: TObject);
var
  i,j,data: integer;
  lSama: boolean;
begin
  randomize;
  j:=0;
  repeat //looping random data integer 1..10
    begin
      data:=RandomRange(1,11);
      lSama:=false;
      for i:=1 to j do
```

```
        begin
          if arrInt[i]=data then
            begin
              lSama:=true;
              break;
            end
        end;
        if not lSama then
          arrInt[j]:=data
        else
          j:=i-1;
        inc(j);
    end
  until j=11;
```

5.6.2.2 Prosedur Simulasi Kartu

Prosedur ini adalah langkah pemanggilan image kartu sesuai dengan nilai array yang diurutkan.

```
procedure TfSorting.simulasi;
begin
  //simulasi semua kartu
  Imaging(Image1,0); // prosedur untuk memanggil masing-masing
                     // image.bmp
  Imaging(Image2,1);
  Imaging(Image3,2);
  Imaging(Image4,3);
  Imaging(Image5,4);
  Imaging(Image6,5);
  Imaging(Image7,6);
  Imaging(Image8,7);
  Imaging(Image9,8);
  Imaging(Image10,9);
end;
```

Prosedur imaging dapat diuraikan sebagai berikut :

```
procedure TfSorting.Imaging(image: TImage; idx: integer);
begin
//procedure pemanggilan gambar ke masing-masing image
  case arrInt[idx] of
    1: image.Picture.LoadFromFile('image\1.bmp');
    2: image.Picture.LoadFromFile('image\2.bmp');
    3: image.Picture.LoadFromFile('image\3.bmp');
    4: image.Picture.LoadFromFile('image\4.bmp');
    5: image.Picture.LoadFromFile('image\5.bmp');
    6: image.Picture.LoadFromFile('image\6.bmp');
    7: image.Picture.LoadFromFile('image\7.bmp');
    8: image.Picture.LoadFromFile('image\8.bmp');
    9: image.Picture.LoadFromFile('image\9.bmp');
    10:image.Picture.LoadFromFile('image\10.bmp');
  end;
```

```
end;
```

5.6.2.3 Prosedur Highlight pada Kode Editor

Prosedur ini berfungsi untuk memberikan sorot (*highlight*) pada baris-baris kode editor sesuai dengan proses yang sedang dieksekusi.

```
procedure TfSorting.SelectMemo(lines: integer);
var
  selcount,i: integer;
begin
  selcount:=0;
  for i:=0 to lines-1 do
     selcount:=selcount+length(Memo1.Lines[i])+2;

  memo1.SetFocus;
  Memo1.SelStart:=selcount;
  Memo1.SelLength:=length(Memo1.Lines[lines]);
end;
```

5.6.2.4 Prosedur mengisi posisi variabel i dan variabel j pada panel variabel

Berfungsi untuk menampilkan letak variabel i dan variabel j pada masing-masing panel variabel.

```
procedure TfSorting.fillvariable(bilangan: integer;varian:string);
begin
  clearVariable(varian);
  case bilangan of
    1: p1.Caption:=varian;
    2: p2.Caption:=varian;
    3: p3.Caption:=varian;
    4: p4.Caption:=varian;
    5: p5.Caption:=varian;
    6: p6.Caption:=varian;
    7: p7.Caption:=varian;
    8: p8.Caption:=varian;
    9: p9.Caption:=varian;
    10:p10.Caption:=varian;
  end;
end;
```

5.6.2.5 Prosedur Pengurutan

Prosedur ini adalah prosedur utama, yaitu prosedur yang berisi langkah-langkah pengurutan sesuai dengan tipe dan metode pengurutan data yang dipilih.

```
procedure TfSor ing.bProsesClick(Sender: TObject);
var
  temp,i,j,k,l,min,max,n:integer;
begin
  n:=10; //jumlah data='0

  case rgTipe.ItemIndex of
  0: begin // tipe ascending (naik)
       case rgMetode.ItemIndex of
         0: begin // metode countsort ascending
              Prosedur countsort_ascending;
            end;
         1: begin //metode selectionsort ascending
              Prosedur selectionsort_ascending;
            end;
         2: begin //metode bubblesort ascending
              Prosedur bubblesort ascending;
            end;
         3: begin // metode insertionsort ascending
              Prosedur insertionsort _ascending;
            end;
    end;
  1: begin //tipe descending  (turun)
       case rgMetode.ItemIndex of
         0: begin //metode countsort descending
              Prosedur countsort_descending;
            end;
         1: begin //metode selectionsort descending
              Prosedur selectionsort _descending;
            end;
         2: begin //metode bubblesort descending
              Prosedur bubblesort _descending;
            end;
         3: begin //metode insertionsort descending
              Prosedur insertionsort _descending;
            end;
         end;
    end;
  end;
end;
```

BAB VI

ANALISIS KINERJA PERANGKAT LUNAK

6.1 Pengujian Program

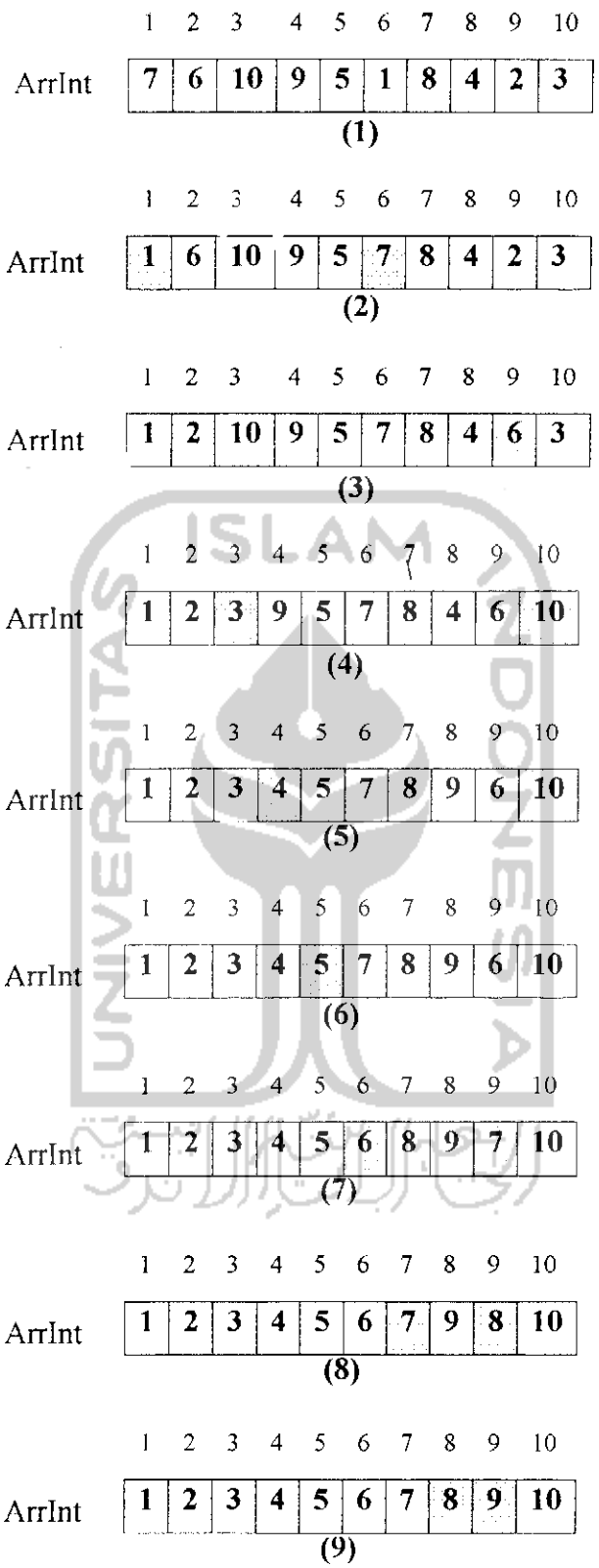
Pemrograman merupakan kegiatan menulis kode program yang akan dieksekusi oleh komputer. Kode program yang ditulis dalam program harus berdasarkan dokumentasi yang disediakan oleh analis sistem, dari hasil rancangan sistem secara rinci. Sebelum progam diterapkan maka program harus bebas dari kesalahan-kesalahan. Oleh karena itu program harus diuji. Pengujian program perlu dilakukan sebelum program tersebut diterapkan ke dalam lingkungan yang sebenarnya. Pengujian tersebut dilakukan untuk menemukan kesalahan-kesalahan yang mungkin terjadi.

6.2 Pengujian dan Analisis

Pada tahap pengujian dan analisis ini membandingkan kebenaran dan kesesuaian dengan kebutuhan sistem. Untuk itu pengujian dilakukan dengan menggunakan proses yang telah valid. Dari tahap pengujian ini akan diketahui kebenaran atau kesalahan proses-proses yang ada dalam program.

6.2.1 Rincian Permasalahan

Dalam rincian permasalahan berikut ini diasumsikan sebuah array (arrInt) yang telah terisi dengan data acak yang akan diurutkan menggunakan metode *selection sort ascending* dengan ilustrasi yang dapat dilihat pada gambar 6.1 :



Gambar 6.1 Ilustrasi Proses Pengurutan Metode *Selection sort* Tipe *Ascending*

	1	2	3	4	5	6	7	8	9	10
ArrInt	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

(10)

	1	2	3	4	5	6	7	8	9	10
ArrInt	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

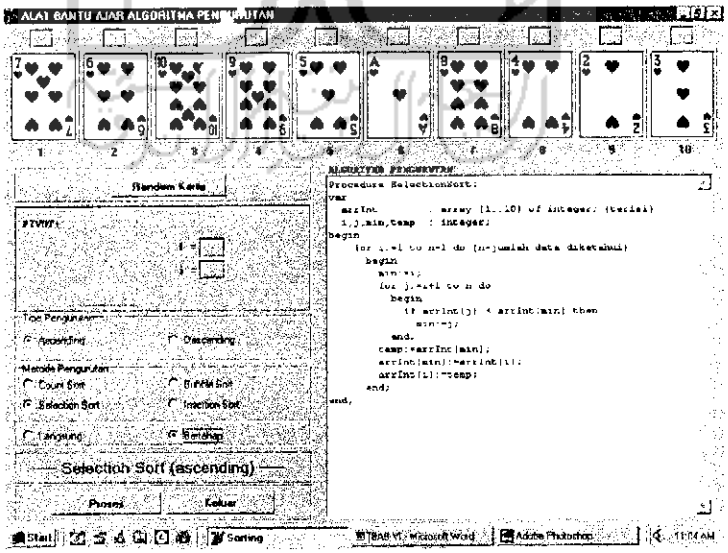
(11)

Gambar 6.1 Lanjutan

Pada gambar di atas dapat dilihat (1) merupakan kondisi awal, langkah (2) sampai langkah (10) merupakan kondisi array pada tiap-tiap langkah pengurutan, dan (11) adalah kondisi akhir dimana array telah terurut.

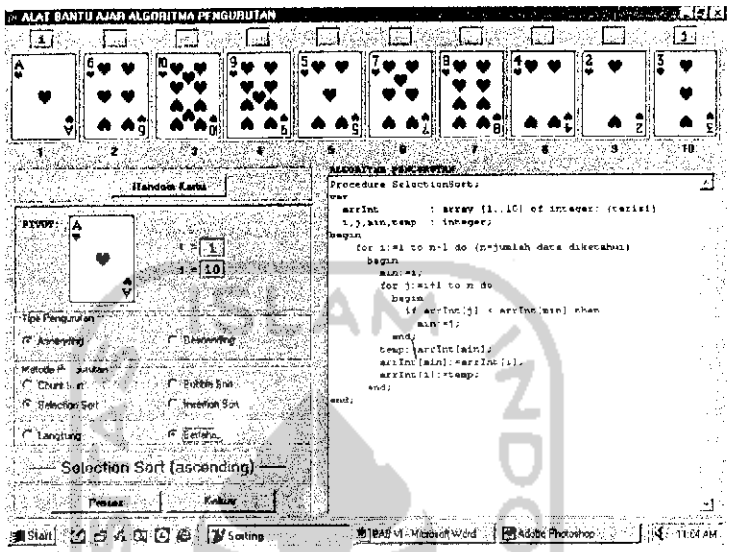
Implementasi pada aplikasi Alat Bantu Ajar Algoritma Pengurutan dapat dilihat pada tiap-tiap tampilan gambar di bawah, yang merupakan langkah-langkah pengurutan pada aplikasi ini.

Pada proses pertama, setelah tombol Random Kartu ditekan maka array akan teracak sekaligus ditampilkan dengan visualisasi kartu seperti gambar 6.2 :



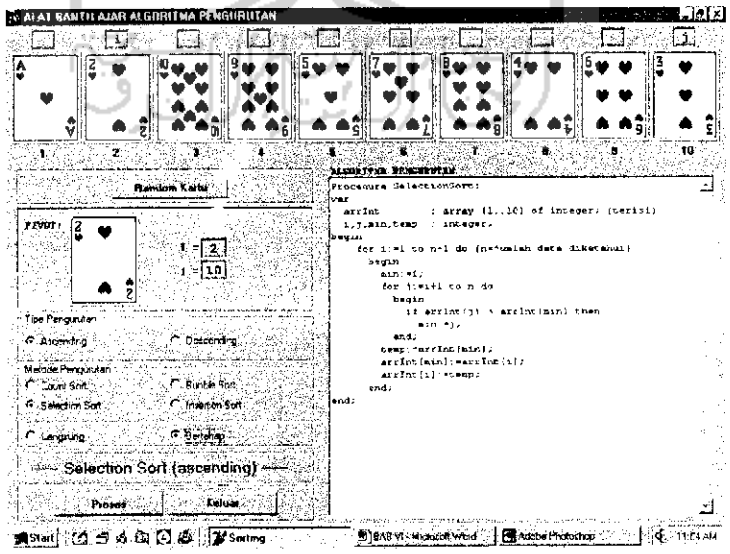
Gambar 6.2 Proses Pertama (Acak Data)

Pada proses kedua kartu 1 telah berpindah pada posisi array ke-1, sedangkan posisi array ke-6 diisi kartu 7, pivot diisi kartu 1. Proses ini dapat dilihat pada gambar 6.3 :



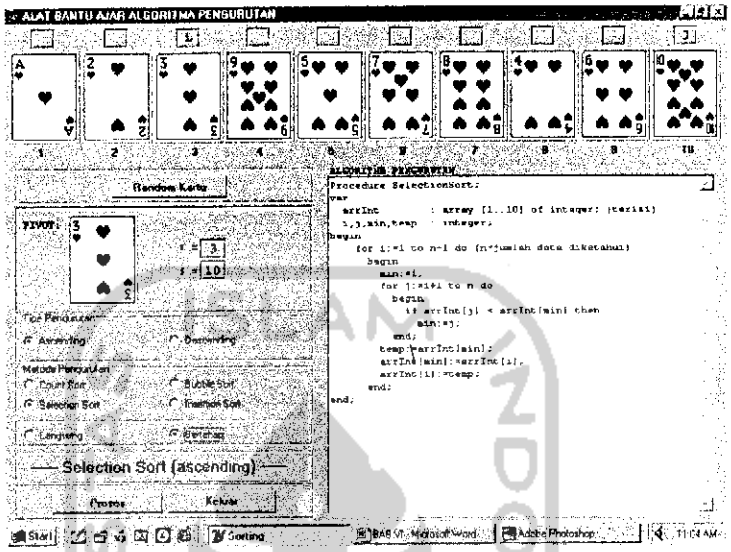
Gambar 6.3 Proses Kedua *Selection Sort Ascending*

Pada proses ketiga kartu 2 telah berpindah pada posisi array ke-2, sedangkan posisi array ke-9 diisi kartu 6, pivot diisi kartu 2. Proses ini dapat dilihat pada gambar 6.4 :



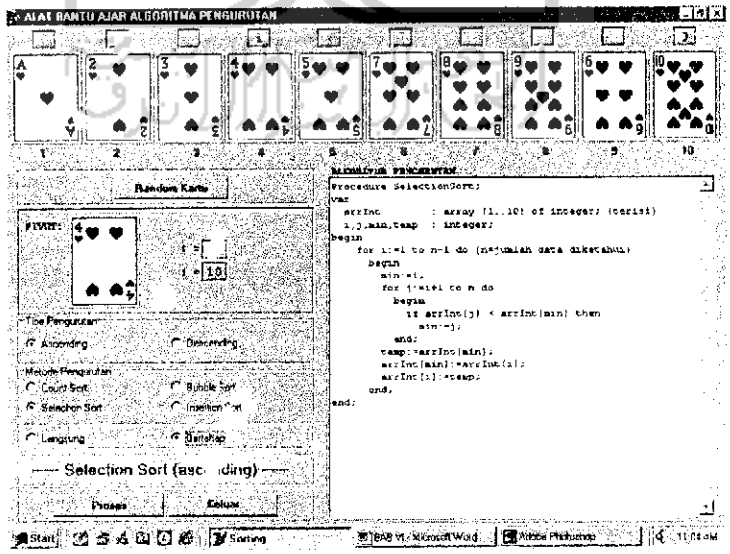
Gambar 6.4 Proses Ketiga *Selection Sort Ascending*

Pada proses keempat kartu 3 telah berpindah pada posisi array ke-3, sedangkan posisi array ke-10 diisi kartu 10, pivot diisi kartu 3. Proses ini dapat dilihat pada gambar 6.5 :



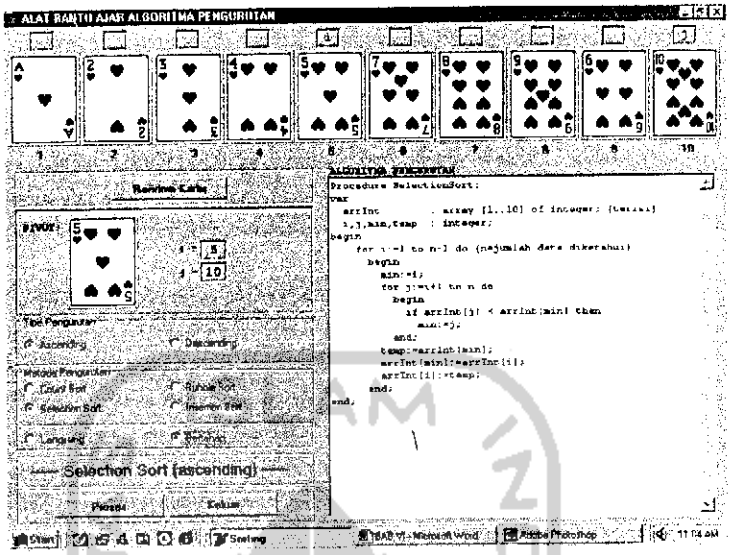
Gambar 6.5 Proses Keempat *Selection Sort Ascending*

Pada proses kelima kartu 4 telah berpindah pada posisi array ke-4, sedangkan posisi array ke-8 diisi kartu 9 , pivot diisi kartu 4. Proses ini dapat dilihat pada gambar 6.6 :



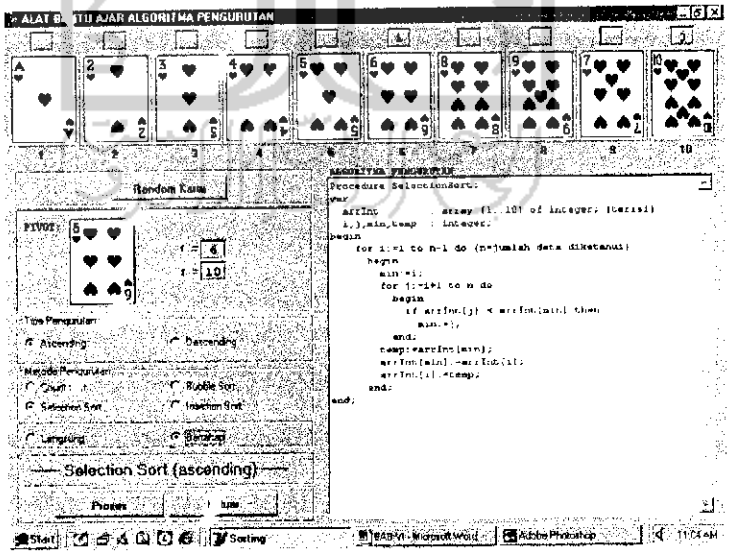
Gambar 6.6 Proses Kelima *Selection Sort Ascending*

Pada proses keenam kartu 5 tetap berada pada posisi array ke-5, pivot diisi kartu 5. Proses ini dapat dilihat pada gambar 6.7 :



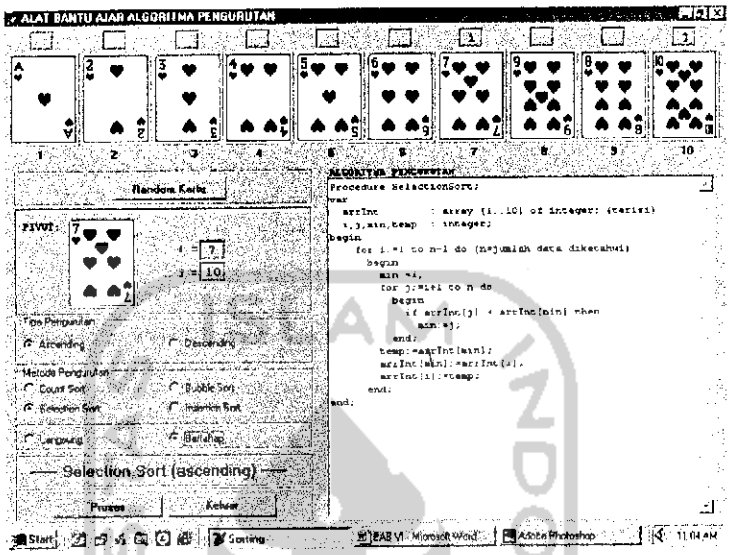
Gambar 6.7 Proses Keenam *Selection Sort Ascending*

Pada proses ketujuh kartu 6 telah berpindah pada posisi array ke-6, sedangkan posisi array ke-9 diisi kartu 7 , pivot diisi kartu 6. Proses ini dapat dilihat pada gambar 6.8 :



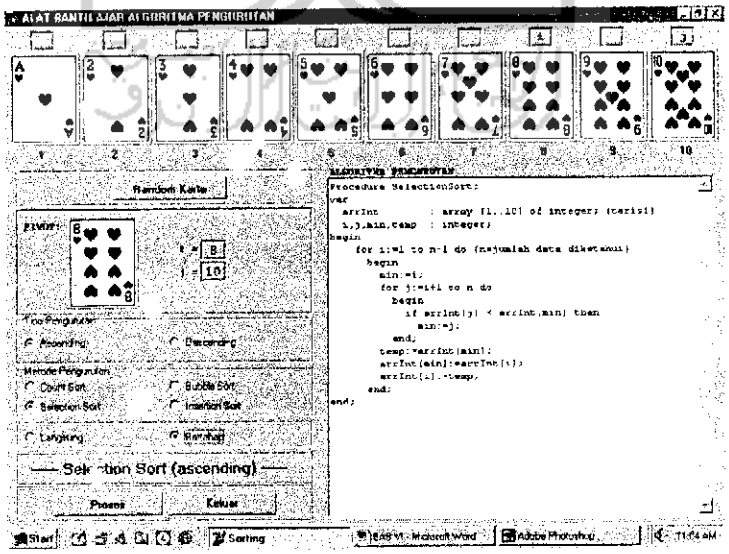
Gambar 6.8 Proses Ketujuh *Selection Sort Ascending*

Pada proses kedelapan kartu 7 telah berpindah pada posisi array ke-7, sedangkan posisi array ke-9 diisi kartu 8 , pivot diisi kartu 7. Proses ini dapat dilihat pada gambar 6.9 :



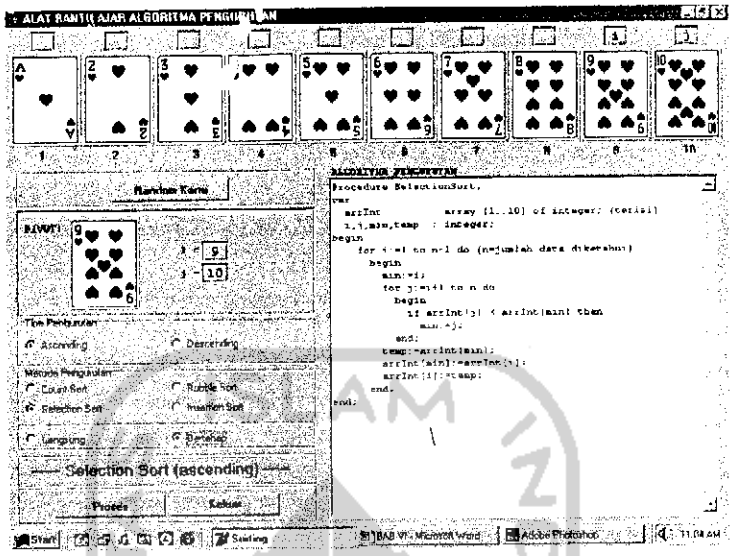
Gambar 6.9 Proses Kedelapan *Selection Sort Ascending*

Pada proses kesembilan kartu 8 telah berpindah pada posisi array ke-8, sedangkan posisi array ke-9 diisi kartu 9 , pivot diisi kartu 8. Proses ini dapat dilihat pada gambar 6.10 :



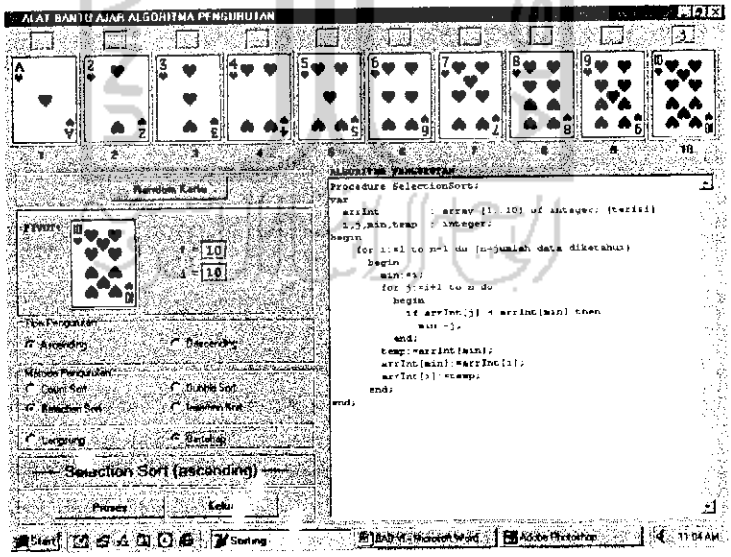
Gambar 6.10 Proses Kesembilan *Selection Sort Ascending*

Pada proses kesepuluh kartu 9 tetap berada pada posisi array ke-9, pivot diisi kartu 9. Proses ini dapat dilihat pada gambar 6.11 :



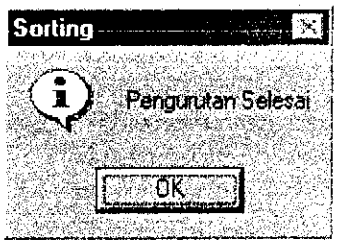
Gambar 6.11 Proses Kesepuluh *Selection Sort Ascending*

Pada proses terakhir semua kartu telah terurut dari nilai terkecil ke nilai terbesar. Proses ini dapat dilihat pada gambar 6.12 :



Gambar 6.12 Proses Kesebelas *Selection Sort Ascending*

Ketika proses pengurutan telah selesai, maka akan ditampilkan pesan bahwa pengurutan telah selesai seperti pada gambar 6.13 :



Gambar 6.13 Pesan pengurutan telah selesai

Pada tampilan kode editor akan ditampilkan *source code* prosedur pengurutan, dan pada saat proses pengurutan dijalankan maka pada kode editor akan ditampilkan highlight pada baris yang sedang dieksekusi seperti pada gambar 6.14 :

Gambar 6.14 Kode Editor Program

BAB VII

KESIMPULAN DAN SARAN

Setelah proses perancangan dan implementasi sistem dilaksanakan dan dilanjutkan dengan analisis kinerja perangkat lunak (pengujian sistem), berikut penulis akan menyimpulkan serta memberikan beberapa saran untuk kesempurnaan sistem ini di masa yang akan datang.

7.1 Kesimpulan

Kesimpulan yang dapat penulis paparkan disini adalah :

1. Aplikasi Alat Bantu Ajar Algoritma Pengurutan ini digunakan untuk mempermudah para siswa dalam mempelajari dan memahami algoritma-algoritma yang diberikan oleh para pengajar.
2. Dengan menggunakan aplikasi Alat Bantu Ajar Algoritma Pengurutan ini, maka para siswa dapat mengetahui proses-proses pengurutan langkah demi langkah yang dapat dilihat pada simulasi kartu. Selain itu dengan aplikasi ini dapat terlihat perbedaan masing-masing metode pengurutan .
3. Penggunaan bahasa pemrograman *Borland Delphi* sangat membantu proses pembuatan aplikasi ini karena *Borland Delphi* sendiri sangat mendukung teknik-teknik pendekatan pemrograman, sehingga memudahkan pembuatan perangkat lunak.

7.2 Saran

Mengingat berbagai keterbatasan yang dialami penulis terutama masalah pemikiran dan waktu, maka hal tersebut berdampak pada kekurangan yang ditemui pada sistem. Untuk itu penulis menyampaikan beberapa saran untuk kesempurnaan sistem ini di masa mendatang.

1. Aplikasi Alat Bantu Ajar Algoritma Pengurutan ini hanya menggunakan beberapa metode pengurutan, untuk pengembangan ke depan dapat ditambah dengan beberapa metode pengurutan yang lain.
2. Jumlah kartu pada aplikasi ini ditentukan sebanyak sepuluh kartu, untuk menambah tingkat fleksibilitasnya maka diharapkan kartu dapat diatur jumlahnya.
3. Penggunaan bahasa pemrograman selain Pascal dapat menambah pengetahuan para siswa yang akan mempelajari algoritma pengurutan.

DAFTAR PUSTAKA

[ALA00] Alam, Agus J. *Borland Delphi 5.0.* Jakarta : Elek Media Komputindo, 2000.

[KAD00] Kadir, Abdul. *Dasar Pemrograman Delphi 5.0 jilid 1.* Yogyakarta : Andi, 2000.

[KAD01] Kadir, Abdul. *Dasar Pemrograman Delphi 5.0 jilid 2.* Yogyakarta : Andi, 2001.

[PRA00] Pranata, Antony. *Algoritma dan Pemrograman.* Yogyakarta : J & J Learning, 2000.

[SUT00] Sutedjo, Budi., dan Michael AN. *Algoritma dan Teknik Pemrograman.* Yogyakarta : Andi, 2000.

[WAH00] Wahid, Fathul. *Dasar-dasar Algoritma dan Pemrograman.* Yogyakarta : Aqila, 2000.

[WIR97] Wirth, Niklaus. *Algoritma + Struktur Data = Program.* Yogyakarta : Andi, 1997.