BAB II

LANDASAN TEORI

2.1 Sistem Perparkiran

Perparkiran termasuk salah satu kegiatan pokok perusahaan yang bergerak dalam bidang jasa transportasi dan saling berhubungan dengan bidang lainnya didalam suatu perusahaan seperti bidang keuangan, bidang perencanaan dan bidang-bidang lainnya dalam usaha untuk mendapatkan keuntungan. Berhasil tidaknya perusahaan dalam mencapai tujuannya, tergantung pada perusahaan tersebut dalam memberikan pelayanan dan pengelolaan serta mengawasi interaksi kegiatan-kegiatan dari masing-masing perusahaan yang bergerak dalam bidang tersebut. Parkir adalah keadaan tidak bergerak suatu kendaraan yang tidak bersifat sementara. Tempat parkir adalah tempat untuk memarkirkan kendaraan. Pengguna parkir adalah pengemudi kendaraan yang menggunakan petak parkir (satuan ruang parkir).

Dalam melakukan kegiatan dibidang jasa perparkiran perlu adanya faktor pendukung, diantaranya :

1. Petugas yang melakukan kegitannya.
2. Komunikasi antara petugas pelayanan jasa parkir dengan pengguna jasa atau pemakai.
3. Faktor-faktor lingkungan seperti tempat yang semakin meningkat merupakan dua hal yang saling mempengaruhi satu sama lain, kebutuhan manusia yang semakin meningkat akan mempengaruhi perkembangan teknologi. Sedangkan

perkembangan teknologi juga akan mempengaruhi pertumbuhan dalam berbagai aspek. Untuk menangani adanya dampak negatif dari adanya teknologi baru sebagai salah satu contoh yaitu pengelolaan sistem perparkiran.

2.2 Sistem Informasi

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu yaitu :

1. Komponen-komponen *(components)*

   Suatu sistem terdiri dari seumlah komponen yang sering disebut dengan sub sistem yang saling berinteraksi artinya saling bekerja sama membentuk satu kesatuan. Sub sistem itu sendiri yang mempunyai sifat-sifat dari sistem itu sendiri dalam menjalankan suatu fungsi dan mempengaruhi proses sistem secara keseluruhan.

2. Batas sistem *(boundary)*

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau yang membatasi tersebut dengan lingkungan luarnya sehingga sistem itu dapat dipandang sebagai satu kesatuan. Batasan sistem dapat merupakan ruang lingkup sistem itu sendiri.

3. Lingkungan hasil sistem *(environments)*

   Lingkungan hasil yaitu segala sesuatu yang ada di luar sistem dari batasan sistem yang bisa mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan juga dapat bersifat merugikan sistem.

4. Penghubung sistem *(interface)*

Penghubung merupakan media penghubung antara komponen sistem sehingga sumber-sumber data mengalir diantara komponen sistem dan membentuk satu kesatuan.

5. Masukkan sistem *(input)*

   Masukkan yaitu energi yang dimasukkan ke dalam sistem.

6. Keluaran sustem *(output)*

   Keluaran yaitu hasil dari energi yang diolah.

7. Pengolah sistem *(process)*

   Proses yaitu yang akan mengubah masukkan menjadi keluaran.

8. Sasaran sistem *(objective)* atau tujuan *(goal)*

   Sistem berguna jika mempunyai sasaran atau tujuan, dan sistem dikatakan berhasil jika keluaran dihasilkan sistem dapat mencapai sasaran dari sistem itu sendiri.

2.2.1 Konsep Dasar Informasi

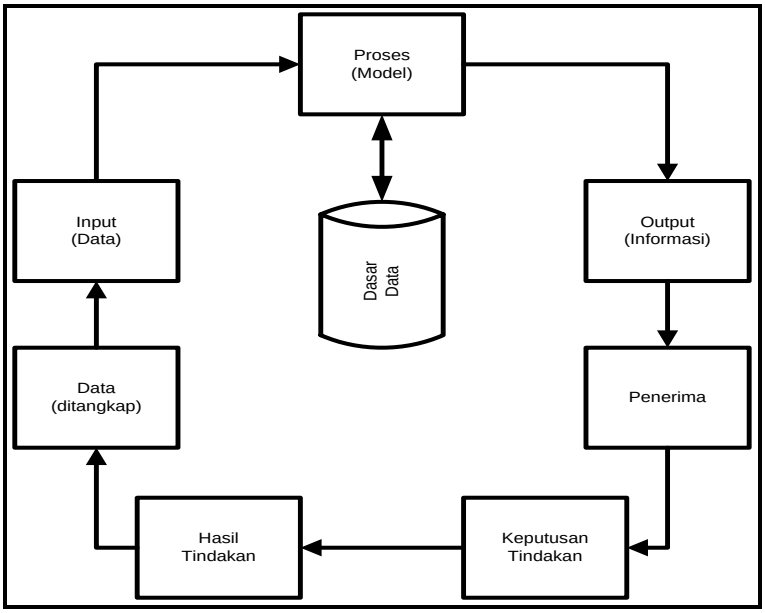
Sumber informasi adalah data. Data merupakan fakta atau kenyataan yang menggambarkan suatu kejadian-kejadian atau event nyata, kemudian dirumuskan ke dalam sekelompok simbol atau lambang-lambang yang teratur yang *menu*njukkan kualitas, tindakan atau hal-hal lain.

Informasi ibarat darah yang mengalir di dalam tubuh suatu organisasi, sehingga informasi sangat penting di dalam suatu organisasi. Karena informasi dapat berguna bagi suatu organisasi atau seseorang dalam mengambil suatu keputusan *menu*rut JOG [6]. Informasi (*information*) dapat didefinisikan sebagai

berikut : *"Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya."*

2.2.2 Siklus Informasi

Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data yang diolah untuk menghasilkan informasi menggunakan suatu model proses yang tertentu. Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai *input*, diproses kembali lewat suatu model dan seterusnya membentuk suatu siklus. Siklus ini oleh John Burch disebut dengan siklus informasi (*information cycle*). Siklus ini disebut juga dengan siklus data (*data processing cycles)*.



Gambar 2.1 Siklus Informasi

2.3 Metode Analisis Sistem Terstruktur

2.3.1 Bagan alir dokumen (*Document flowmap*)

Bagan alir dokumen menggambarkan aliran dokumen dan informasi antar area pertanggungjawaban didalam sebuah organisasi. Bagan alir ini menelusuri sebuah dokumen dari asalnya sampai tujuannya. Secara rinci bagan alir ini *menu*njukan dari mana dokumen berasal, didistribusikannya, tujuan digunakan dokumen tersebut. Bagan alir ini bermanfaat untuk menganalisis kecukupan prosedur pengawasan dalam sebuah sistem. Bagan alir dokumen disebut juga bagan alir formulir yang *menu*njukan arus dari laporan dan formulir termasuk tembusannya.

2.3.2 *Entity Relationship Diagram* (ERD)

ERD merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan. ERD digunangan untuk memodelkan struktur data dan hubungan antar data, karena hal ini relatif kompleks. Dengan ERD kita dapat menguji model dengan mengabaikan proses yang dilakukan.

ERD menggunakan Sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data, pada dasarnya ada 3 macam simbol yang digunakan:

1. *Entity*

   Entity adalah suatu objek yang dapat di identifikasi dalam lingkaran pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat.

2. Atribut

   Entity mempunyai elemen yang disebut atribut dan berfungsi mendeskripsikan karakter *entity*.

3. Relasi

   Relasi merupakan gugusan entitas yang berhubungan antar entitas atau beberapa entitas. Macam-macam relasi :

   a. Relasi satu ke satu (*one to one*)

      Setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B, dan berlaku sebaliknya.

   b. Relasi satu ke banyak (*one to many*)

      Setiap entitas pada himpunan A berhubungan dengan banyak entitas pada himpunan entitas B, dan B berhubungan paling banyak dengan satu entitas pada himpunan entitas A.

   c. Relasi banyak ke satu (*many to one*)

      Setiap entitas pada himpunan a berhubungan paling banyak dengan satu entitas pada himpunan entitas B, tapi tidak sebaliknya dimana setiap entitas di himpunan A berhubungan paling banyak satu entitas pada himpunan entitas.

   d. Relasi banyak ke banyak (*many to many*)

      Setiap entitas pada himpunan A berhubungan paling banyak dengan banyak entitas pada himpunan entitas.

2.3.3 Diagram Konteks (*Contexts diagram*)

Diagram konteks ini merupakan alat-alat untuk struktur analisis. Pendekatan struktur ini mencoba untuk menggambarkan sistem secara garis besar atau secara keseluruhan. Diagram konteks ialah kasus khusus dari DFD atau bagian dari DFD yang berfungsi memetakan modul lingkungan yang direpresentasikan dengan lingkaran yang mewakili keseluruhan sistem.

Diagram Kontek meliputi beberapa sistem antara lain :

1. Kelompok pemakai
2. Data yang diterima oleh sistem dari lingkaran
3. Data yang dihasilkan oleh sistem
4. Penyimpanan data

2.3.4 *Data Flow Diagram* (DFD)

DFD adalah penjelasan lebih rinci dari diagram konteks dan proses fungsional yang ada dalam sistem. DFD mejelaskan tentang aliran masuk, aliran keluar, proses serta penyuntingan *file* yang digunakan.

DFD digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang dikembangkan secara logika tanpa memperhatikan lingkungan fisik dimana data tersebut mengalir atau disimpan.

DFD sangat berguna untuk mengetahui prosedur suatu program. Keuntungan yang lain adalah mempermudah pemakai atau *user* yang kurang menguasai komputer untuk mengerti sistem yang akan dibuat.

2.3.5 Kamus Data *(Data directory)*

*Menu*rut JOG[6] mendefinisikan kamus data adalah sebagai berikut :

"Kamus data atau data directory adalah catalog data tentang fakta dan kebutuhan-kebutuhan informasi dari suatu sistem informasi"

Dengan menggunakan kamus data, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Pada tahap perancangan sistem, kamus data dapat digunakan untuk merancang *input*, *output* (laporan-laporan) dan merancang *database* program.

2.4 Basis Data

Fabbri dan A. Robert Schwab, mendefinisikan basis data sebagai berikut :

"*Basis data adalah sistem berkas terpadu yang dirancang terutama untuk meminimalkan pengulangan data"* [6JOG].

*Menu*rut George Tsu-der Chou, basis data dapat didefinisikan sebagai berikut :

"Basis data sebagai kumpulan informasi bermanfaat yang diorganisasikan ke dalam tatacara yang khusus". [6JOG]

Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas. Sistem basis data adalah suatu sistem menyusun dan mengelola *record* menggunakan komputer untuk menyimpan atau merekam serta memelihara data opersional lengkap sebuah organisasi / perusahaan sehingga mampu menyediakan informasi yang optimal yang diperlukan pemakai untuk proses mengambil keputusan.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut DBMS. DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien.

Mengapa diperlukan *database* :

1. Salah satu komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
2. Menentukan kualitas informasi : akurat, tepat pada waktunya dan relevan.
3. Informasi dapat dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkanya.
4. Mengurangi duplikasi data (data *redudancy*).
5. Hubungan data dapat ditingkatkan.
6. Mengurangi pemborosan tempat simpanan luar.

Sampai dengan membentuk suatu *database*, data mempunyai jenjang mulai dari karakter-karakter, item data, *record*, *file* dan kemudian *database*. Jenjang data dapat digambarkan sebagai berikut :

1. *Character* : merupakan bagian data yang terkecil, dapat berupa karakter *numeric*, huruf ataupun karakter-karakter khusus (*special character*) yang mrmbentuk suatu *field*.
2. *Field* : merepresentasikan suatu atribut dari *record* yang *menu*njukan suatu item dari data, seperti misalnya nama, alamat dan lain sebagainya. Kumpulan dari *field* membentuk *record*.
3. *Record* : kumpulan dari *field* membentuk suatu *record*. *Record*

menggambarkan suatu unit data individu yang tertentu. Kumpulan dari *record* membentuk suatu *file*.

4. *File* : *file* terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis.
5. *Database* : kumpulan dari *file* yang membentuk suatu *database*.

2.4.1 Tahap Perancangan Basis Data

Perancangan basis data merupakan langkah untuk menentukan basis data yang diharapkan dapat mewakili seluruh kebutuhan pengguna. Perancangan basis data terdiri atas perancangan basis data secara konseptual, perancangan basis data secara logis, dan perancangan basis data secara fisis.

Beberapa komponen yang terdapat pada perancangan basis data secara konseptual antara lain:

1. Entitas

   Entitas terkadang disebut tipe entitas atau kelas entitas. Entitas adalah objek yang dapat dibedakan dari objek-objek lainnya.

2. Atribut

   Atribut adalah item data yang menjadi bagian dari suatu entitas. Istilah lain dari atribut adalah properti.

3. Hubungan

   Hubungan adalah asosiasi atau kaitan antara dua entitas.

4. Kekangan

   Kekangan digunakan untuk melindungi integritas data (misalnya, melindungi

kesalahan sewaktu pengisian data).

5. *Domain*

   Domain adalah himpunan yang berlaku bagi suatu atribut. Domain mendefinisikan nama, tipe, format, panjang, dan nilai masing-masing *item* data.

6. *Integritas Referensial*

   Integritas referensial adalah aturan-aturan yang mengatur hubungan antara kunci primer dengan kunci tamu milik tabel-tabel yang berbeda dalam suatu basis data relasional untuk menjaga konsistensi data.

2.5 Teknik Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan mempesentasikan kajian pokok dari spesifikasi, desain, dan pengkodean.

2.5.1 Dasar Pengujian Perangkat Lunak

Pengujian menyajikan *anomali* yang menarik bagi perekayasa perangkat lunak. Pada proses perangkat lunak, perekayasa pertama-tama berusaha membangun perangkat lunak dari konsep abstrak ke implementasi yang dapat dilihat, baru dilakukan pengujian. Perekayasa menciptakan sederetan *test case* yang dimaksudkan untuk "membongkar" perangkat lunak yang sudah dibangun. Pada dasarnya pengujian merupakan salah satu langkah dlam proses rekayasa perangkat lunak yang dianggap sebagai hal yang destruktif daripada konstruktif.

2.5.2 Sasaran-sasaran Pengujian

Sejumlah aturan yang berfungsi sebagai sasaran pengujian:

1. Pengujian adalah proses eksekusi suatu program dengan meksud menemukan kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

2.5.3 Prinsip Pengujian

Sebelum mengaplikasikan metode untuk mendesain *test case* yang efektif, perekayasa harus memahami prinsip dasar yang *menu*ntun pengujian perangkat lunak diantaranya:

1. Semua pengujian harus dapat ditelusuri sampai kepersyaratan pelanggan.
2. Pengujian harus direncanakan lama sebelum pengujian itu mulai.
3. Prinsip pareto berlaku untuk pengujian perangkat lunak.
4. Pengujian harus mulai dari yang kecil dan yang berkembang kepengujian yang lebih besar.
5. Pengujian yang mendalam tidak mungkin.
6. Untuk menjadi paling efektif pengujian harus dilakukan oleh pihak ketiga yang *independen.*

2.5.4 Testabilitas

Testibilitas perangkat lunak adalah seberapa mudah program komputer dapat diuji. Karena pengujian sulitk, maka perlu diketahui apa yang harus dilakukan agar manjadi lebih mudah. *Cheklist* berikut ini memberikan serangkaian karakteristik yag membawa peragkat lunak yang dapat diuji.

1. *Operabilitas,* "semakin baik dia bekerja, semakin efisien dia diuji".
2. *Observabilitas,* "apa yanganda lihat adalah apa yang anda uji".
3. *Kontrabilitas,* semakin baik kita dapat mengontrol perangkat lunak semakin banyak pengujian yang diotomatisasi dan dioptimalkan".
4. *Dekomposabilitas,* "dengan mengontrol ruang lingkup pengujian, kita dapat lebih cepat mengisolasi masalah dan melakukan pengujian kembali secara lebih halus".
5. Kesederhanaan, "semakin cepat yang diuji, semakin sedikit kita dapat mengujinya".
6. *Stabilitas,* "semakin sedikit perubahan, semakin gagguan dalam pengujian".
7. *Verifikasi,* mengacu kepada rangkaian aktivitas yang memastikan bahwa perangkat lunak secara tepatmengimplementasikan suatu fungsi terentu.
8. *Validasi,* mengacu pada rangkaian aktivitas berbeda yang memastikan bahwa prangkat lunak yang dibangun dapat ditelusuri kepersyaratan pelanggan. "Apakah kita membangun produk yang benar".

2.5.5 Pengujian *Black Box*

Pengujian *Black-box* berfokus pada persyaratan fungsional perangkat

lunak. Dengan demikian, pengujian *black-box* memungkinkan perekayasa peangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya semua persyaratan fungsional untuk suatu program.

Penguian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *Interface.*
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

2.6 Sistem *Client Server*

Sistem *client-server* mempunyai dua komponen utama yaitu komputer *client* dan komputer *server*. *Server* merupakan komputer induk yang melakukan pemprosesan terbanyak untuk me*menu*hi permintaan-permintaan dari komputer *client* dan bertindak sebagai *server database* yang menyimpan data. *Client* yaitu suatu komputer atau *workstation* yang melakukan pengiriman permintaan-permintaan data pada *server* kemudian menampilkan data tersebut pada *interface* aplikasi yang dimilikinya. Selain itu *client* juga mempunyai kemampuan untuk mengubah atau menghapus data.

Sistem *client-server* merupakan suatu sistem *client* komputer yang melibatkan proses-proses *client* yang meminta suatu pelayanan data kepada komputer *server* yang menyediakan layanan tersebut, sehingga *client* maupun

server sama-sama melakukan pekerjaan. Dengan adanya kombinasi *client* dan *server* ini maka kumpulan dari modul-modul program tidak dieksekusi dalam memori yang sama namun terbagi dalam komputer *client-server*.

Hal ini menjadikan konfigurasi bagi komputer *client* dan komputer *server* bisa berbeda seperti kapasitas memori, kecepatan prosesor atau alat masukan dan keluaran yang disesuaikan dengan fungsi kerja dari elemen-elemen tersebut. Bagi *server* yang menjalankan tugas pengelolaan suatu *database* digunakan suatu konfigurasi yang khusus menangani tugasnya tersebut dengan sistem operasi yang dikhususkan bagi *server* seperti *windows NT server, windows 2000 server*, sedangkan komputer *client* menggunakan konfigurasi yang umum bagi sebuah komputer dekstop yang terhubung ke jaringan dengan sistem operasi seperti *windows 98, windows me, windows xp* dan lain-lain.

2.6.1 Cara Kerja *Client-Server*

Sistem *client-server* berjalan seutuhnya pada dua sistem yang berbeda. Biasanya sebuah *server* melayani satu *client* saja. Apabila pemakainnya mengakses informasi bagian aplikasi, *client* mengeluarkan permintaan yang dikirimkan melalui jaringan kepada *server*. *Server* kemudian menjalankan permintaan dan mengirimkan kembali kepada *client.*

Proses *server* berperan sebagai aplikasi yang mengelola sumber daya nilai bersama (*shared resource*) seperti *database*, printer atau jalur komunikasi menjalankan tugasnya. Sebagai *back-end*, sistem *client-server* yaitu pusat pemprosesan data, sedangkan proses *client* meliputi program-program untuk

mengirimkan permintaan pada *server* serta melakukan pengaksesan pada data seperti mengubah, menghapus atau menambah data.

Karena itu program pada *client* adalah aplikasi *front-end* yang digunakan sebagai antarmuka bagi pemakai untuk berinteraksi dengan *server* selain itu *client* menangani pemakaian sumber daya lokal seperti *monitor*, *keyboard* dan perangkat lokal lainya.

2.7 *Software* Pendukung

Untuk perangkat lunak pendukung cara mengembangkannya melalui dua macam aplikasi, karena komputer *server* dan *client* memerlukan aplikasi yang berbeda. *Operating system* komputer menggunakan *windows* SP 2, My SQL digunakan untuk pengelolaan *database server* dan Borland Delphi sebagai aplikasi pemograman untuk mengembangkan aplikasi sistem *client-servet*. Kedua perangkat lunak ini dinilai merupakan perangkat lunak yang digunakan sebagai alat yang mengembangkan aplikasi bagi sistem *client-server*.

2.7.1 Borland Delphi 7.0

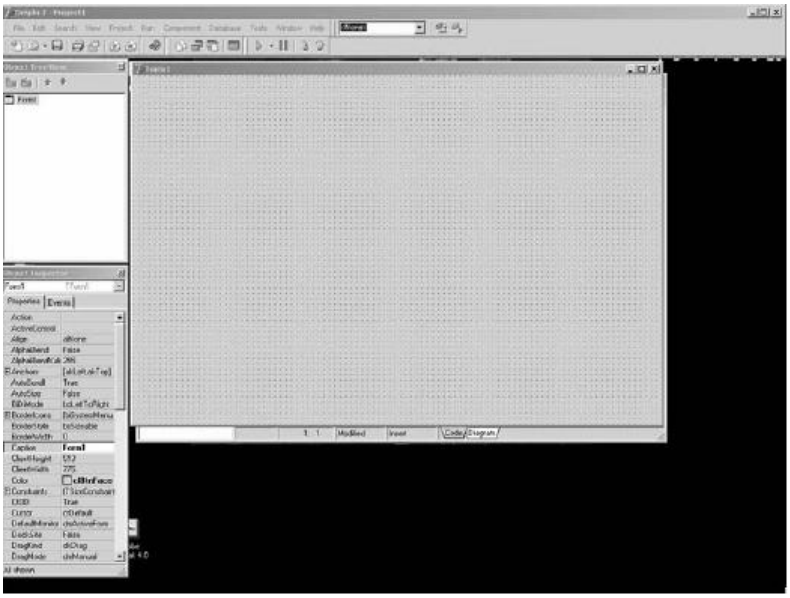
Delphi adalah *komplier* atau penerjemah bahasa Delphi (awalnya dari bahasa pascal) yang merupakan bahasa tingkat tinggi. Bahasa pemograman di Delphi disebut bahasa *procedural* artinya bahasa atau sintaknya mengikuti urutan tertentu atau prosedur. Ada jenis pemograman non-prosedural seperti pemograman untuk kecerdasan buatan seperti bahasa prolog. Delphi termasuk keluarga visual basic, visual C, artinya perintah-perintah untuk membuat objek

dapat dilakukan secara visual. Pemogram hanya memilih objek apa yang ingin dimasukan kedalam *form*, kemudian tingkah laku objek tersebut akan menerima aksi tinggal dibuat programnya. Delphi merupakan bahasa berorientasi objek, artinya nama objek, *property* dan prosedur dikemas menjadi satu kemasan (*encapsulate*).

Delphi adalah sebuah perangkat lunak untuk membuat aplikasi komputer berbasis windows. Delphi merupakan bahasa pemograman berbasis objek, artinya semua komponen yang ada merupakan objek-objek.ciri-ciri sebuah objek adalah memlki nama, *property* dan prosedur. Delphi disebut juga *visual programming* artinya komponen-komponen yang ada tidak hanya berupa teks tetapi muncul berupa gambar-gambar.

2.7.1.1 Mengenal IDE Borland Delphi 7

Lingkungan pengembangan terpadu atau *Integrated Development Environment* (IDE) dalam Borland Delphi terbagi menjadi delapan bagian utama, yaitu *Main Window, Toolbar, Component Palette, Form Designer, Code Editor, Object Inspector, Code Explorer, dan Object TreeView* (Madcoms.2003).



Gambar 2.2 Lembar Kerja Borland Delphi

1. *Main Window*

   Jendela utama adalah bagian IDE yang mempunyai fungsi yang sama dengan semua fungsi utama dari program aplikasi windows lainnya, jendela utama Delphi dibagi menjadi tiga bagian, berupa *Main Menu, Toolbar,* dan *Component Palette.*

2. *Main Menu*

   Menu Utama pada Delphi memiliki kegunaan yang sama seperti program aplikasi windows lainnya. Dengan menggunakan fasilitas *menu*, kita dapat menyimpan atau memanggil program. Pada dasarnya perintah dapat ditemukan pada bagian *menu* utama.

3. *Toolbar*

   Delphi memiliki beberapa *Toolbar* yang memiliki perbedaan fungsi dan setiap tombol pada bagian *toolbar* berfungsi sebagai pengganti suatu *menu* perintah yang sering digunakan. *Toolbar* sering disebut

juga dengan *Speedbar*. *Toolbar* terletak pada bagian bawah baris *menu*. Pada kondisi *default* Delphi memiliki enam bagian toolbar, antara lain: *Standard, View, Debug, Desktops, Custom* dan *Component Palette.*



Gambar 2.3 *Toolbar (speedbar)*

4. *Component Palette*

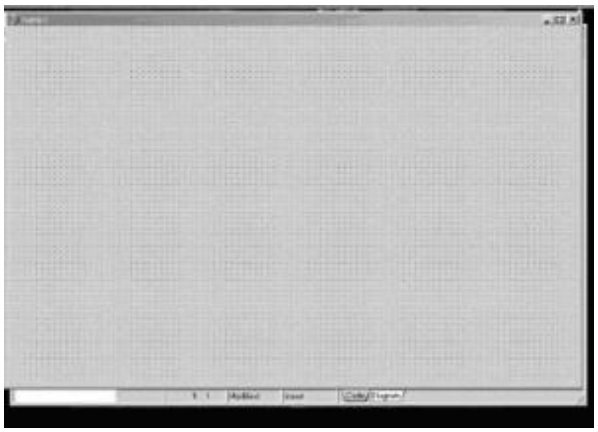
   Component Palette berisi kumpulan ikon yang melambangkan komponen-komponen yang terdapat pada VCL (*Visual Component Library*).



Gambar 2.4 *Component Palette*

5. *Form Designer*

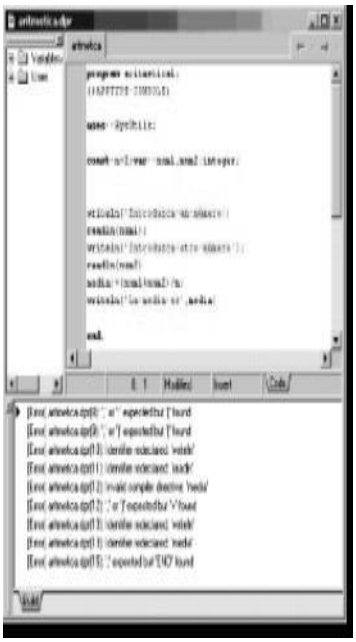
   Form Designer merupakan suatu objek yang dapat dipakai sebagai tempat untuk merancang program aplikasi. Form berbentuk sebuah meja kerja yang dapat diisi dengan komponen-komponen yang diambil dari *component palette.*



Gambar 2.5 *Form Designer*

6. *Code Editor*

Code Editor merupakan tempat di mana *menu*lis kode program.



Gambar 2.6 *Code editor*

7. *Object Inspector*

Object Inspector digunakan untuk mengubah *property* atau karakteristik dari sebuah komponen. *Object inspector* terdiri dari dua tab, yaitu *Properties dan events.*



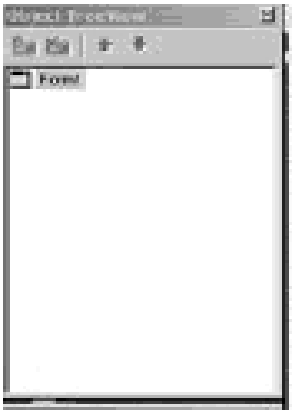
Gambar 2.7 *Object inspector*

8. *Code Explorer*

   Jendela *Code Explorer* adalah lembar kerja baru yang terdapat di dalam Delphi 7 yang tidak ditemukan pada versi-versi sebelumnya. *Code explorer* digunakan untuk memudahkan pemakai berpindah antar *file* unit yang terdapat di dalam jendela *code editor*.

9. *Object TreeView*

   Object TreeView menampilkan diagram pohon dari komponen-komponen yang bersifat visual maupun nonvisual yang telah terdapat dalam *form*, data *module*, atau *frame*. *Object TreeView* juga menampilkan hubungan logika antar komponen.



Gambar 2.8 *Object TreeView*

2.7.1.2 Komponen Borland Delphi 7

Komponen-komponen Borland Delphi 7 [KUS4) :

1. *Project*

   Project adalah sekumpulan *form*, *unit* dan beberapa hal lain singkatnya *project* adalah program aplikasi itu sendiri. *File-file* yang terdapat pada *Hard Disk* saat proses kompilasi proyek pada Delphi 7.0 yaitu : *file project* (.dpr), *file unit* (.pas), *file form* (.dfm), *file* dcu, *file resource* (.res), *file* exe (merupakan *file* hasil dari pengkopilasian *file project*), *file back-up* (.~dp, ~df, .~pa), dan *file project option* (.dfo).

2. *Form*

   Form adalah suatu objek yang dipakai sebagai tempat bekerja program aplikasi. Setiap *form* mengandung *unit*. *Unit* dalam *form* dipakai untuk mengatur dan mengendalikan *form* serta untuk berinteraksi dengan komponen lain.

3. *Unit*

Unit adalah modul kode program. Dalam Borland Delphi 7.0, ada *unit* yang tak terpisahkan dengan *form*. Setiap kali dibuat satu *form*, maka otomatis pula dibuat satu *unit*. *Unit* yang berhubungan dengan *form* dipakai untuk mengatur dan mengendalikan sesuatu yang berhubungan dengan *form* dan berinteraksi dengan komponen lainnya.

Manfaat menggunakan unit antara lain sebagai berikut :

Membagi program aplikasi yang besar dalam beberapa unit, sehingga kita dapat mengedit unit-unit tertentu saja.

Membuat *library* (daftar pustaka) berupa *function* dan *procedure*, sehingga memudahkan sharing antarprogram.

Unit dapat dikompilasi terpisah dari program aplikasi, sehingga program aplikasi lain yang memerlukan *unit* serupa dapat menggunakannya tanpa harus *menu*lis kembali kode programnya.

4. *Program*

   Program dibangun dari satu atau lebih *unit*, sebuah *program*, secara umum mempunyai stuktur sebagai berikut :

   a. *Heading program*, yaitu bagian yang *menu*njukan nama *program* tersebut.
   b. Pernyataan *uses*, yaitu berisi daftar *unit* yang dipakai *program*.
   c. Blok deklarasi dan pernyataan, yaitu bagian yang berisi deklarasi dan pernyataan *program* yang dilaksanakan pada saat *program* dijalankan.

5. *Property*

Property Digunakan untuk menentukan *setting* suatu objek. Suatu objek mempunyai property yang dapat diatur dari *properties* dalam jendela *object* inspector maupun siatur lewat kode program.

6. *Event*

   Event adalah peristiwa atau kejadian yang diterima oleh suatu objek.

7. *Method*

   Method adalah *procedure* atau perintah yang melekat pada suatu object. *Method* hampir sama dengan *property*, bedanya adalah *property* dipakai untuk menampung dan mengambil suatu nilai, sedangkan *method* untuk melakukan aksi.

2.8 *Database*

Khusus untuk *database*, Borland Delphi menyediakan fasilitas objek yang kuat dan lengkap yang memudahkan dalam membuat program. tipe *database* yang dimiliki Delphi adalah tipe *database Paradox, dBase, MS.Access, ODBC, SyBase,Oracle, FoxPro, MS SQL Server, dan Interbase.*

Salah satu manfaat *database* yang paling utama adalah untuk memudahkan dalam mengakses data. Kemudahan dalam mengakses data ini adalah sebagai implikasi dari keteraturan data yang merupakan syarat mutlak dari suatu *database* yang baik.

Dalam *database* semakin teraturnya dalam menyimpan maka akan semakin mudah dalam melakukan pengaksesan. Semakin teratur penyimpanan semakin

banyak usaha yang harus dilakukan. Dengan kata lain keteraturan berbanding lurus dengan kesulitan. Namun dari keteraturan akan memberikan banyak keuntungan, diantaranya adalah kemudahan dalam hal pengaksesannya.

Membangun *database* adalah langkah awal dari pembuatan sebuah aplikasi. Keberhasilan dalam membangun *database* akan menyebabkan program lebih mudah dibaca, mudah dikembangkan dan mudah mengikuti perkembangan perangkat lunak. Komponen dasar suatu *database* terdiri dari :

1. Tabel, adalah kumpulan dari suatu *field* dan *record*. Dalam hal ini biasanya *field* ditunjukan dalam bentuk kolom dan *record* ditunjukan dalam bentuk baris.
2. *Field,* adalah sebutan untuk mewakili suatu *record*. Misalnya seorang pegawai dapat dilihat datanya melalui *field* yang diberikan padanya seperti nip, nama, alamat, dan lain-lain.
3. *Record,* adalah kumpulan elemen-elemen yang saling berkaitan. menginformasikan tentang suatu isi data secara lengkap. Satu *record* mewakili satu data atau atau informasi tentang seseorang misalnya, nomor daftar, nama pendaftar, alamat, tanggal masuk.
4. *Primary Key,* adalah suatu kolom (*field*) yang menjadi titik acuan pada sebuah tabel, bersifat unik dalam artian tidak ada satu nilaipun yang sama atau kembar dalam tabel tersebut, dan dalam satu tabel hanya boleh ada satu *primary key*.
5. *Foreign Key* atau disebut juga kunci relasi adalah suatu kolom dalam *table* yang digunakan sebagai “kaitan” untuk melengkapi satu hubungan yang

didapati dari tabel induk, dan biasanya hubungan yang terjalin antar *table* adalah satu ke banyak (*one to many*).

6. *Index*, adalah struktur basis data secara fisik, yang digunakan untuk optimalisasi pemrosesan data dan mempercepat proses pencarian data.

2.9 Kode Baris (*Barcode*)

Kode baris digambarkan dalam bentuk baris hitam tebal dan tipis yang disusun berderet sejajar horisontal. Untuk membantu pembacaan secara manual dicantumkan juga angka-angka dibawah kode baris tersebut. Angka-angka tersebut tidak mendasari pola kode baris yang tercantum. Ukuran dari kode baris tersebut dapat diperbesar maupun diperkecil dari ukuran nominalnya tanpa tergantung dari mesin yang membaca.

2.9.1 *Barcode* Satu Dimensi (1D)

Barcode satu dimensi biasanya dinamakan *linear bar codes* (kode berbentuk baris). Contoh *barcode* satu dimensi adalah sebagai berikut :

1. *Code* 39 (*code* 3 of 9)

   Adalah sebuah *barcode alphanumerik* (full ASCII) yang memiliki panjang baris yang bervariasi. Aplikasi *barcode* jenis *code* 39 adalah untuk *inventory*, *asset tracking* dan digunakan pada tanda pengenal identitas.



Gambar 2.9 *Barcode* jenis *Code* 39

2. *Cod*e 128

   Adalah suatu *barcode alphanumerik* (full ASCII) yang memiliki kerapatan (*density*) yang sangat tinggi dan panjang baris yang bervariasi. *Barcode code* 128 ideal untuk aplikasi seperti *shipping and warehouse management* (pangaturan maskapai pelayaran dan pengelolaan gudang).



Gambar 2.10 *Barcode* jenis *Code* 128

3. *Interleaved* 2 of 5

   Adalah sebuah *barcode* yang berbentuk numerik dan memiliki panjang baris yang bervariasi. *Barcode interl*eaved 2 of 5 dapat dipergunakan untuk aplikasi industri dan laboratorium.



Gambar 2.11 *Barcode* jenis *Interleave*d 2 of 5

4. UPC (*Universal Product Code*)

   Adalah sebuah *barcode* yang berbentuk numerik dan memiliki panjang baris yang tetap (*fixed*). UPC digunakan untuk pelabelan pada produk-produk kecil/eceran (*retail product labeling*). Simbol ini dibuat untuk kemudahan pemeriksaan keaslian suatu produk. Bilangan-bilangan UPC harus diregistrasikan atau terdaftar di *Uniform Code Council*.



Gambar 2.12 *Barcode* jenis UPC

2.9.2 *Barcode* Dua Dimensi (2D)

Adalah *barcode* yang dikembangkan lebih dari sepuluh tahun lalu, tetapi baru sekarang ini mulai semakin populer. *Barcode* dua dimensi ini memiliki beberapa keuntungan dibandingkan *linear bar codes* (*barcode* satu dimensi) yaitu, dengan menggunakan *barcode* dua dimensi, informasi atau data yang besar dapat



Gambar 2.13 *Barcode* jenis PDF417

disimpan di dalam suatu ruang (*space*) yang lebih kecil. Contoh *barcode* dua dimensi adalah "symbology PDF417" yang dapat menyimpan lebih dari 2000 karakter di dalam sebuah ruang (*space*) yang berukuran 4 *inch* persegi (in^2).

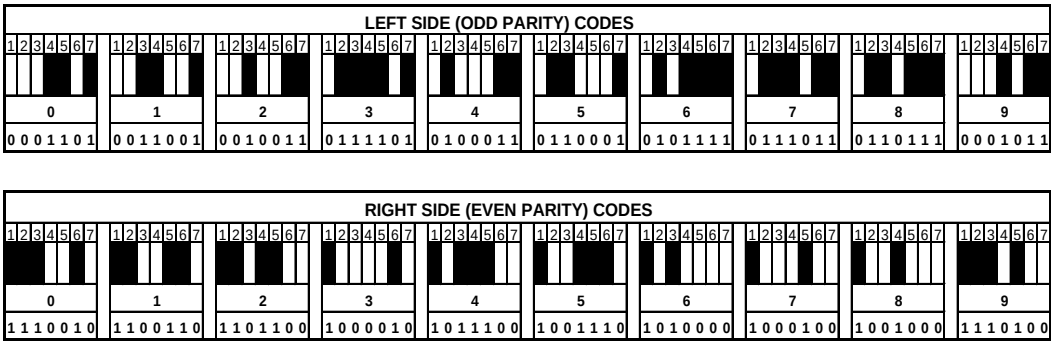
2.9.3 Cara *Scanner* Membaca *Barcode*

Suatu bilangan *barcode* tunggal sebenarnya terdiri dari tujuh unit. Satu unit terdiri dari salah satu warna hitam atau putih. Sebuah unit yang berwarna hitam ditunjukkan dengan sebuah ***bar***, sedangkan yang berwarna putih ditunjukkan dengan sebuah ***space*** (spasi). Cara lain penulisan *barcode* adalah dengan bilang "1" untuk menyatakan ***black bar*** dan bilangan "0" untuk

menyatakan ***white space***. Misalnya, tujuh unit berikut ini adalah 0011001 dapat dinyatakan sebagai berikut *space-space-bar-bar-space-space-bar*. Sebuah *barcode* UPC bilangan di sisi bagian kiri *barcode* (kode perusahan/manufaktur) dikodekan berbeda dengan bilangan di sisi bagian kanan (kode produk). Bilangan yang berada sebelah kiri merupakan kebalikan dari bilangan yang ada di sebelah kanan, misalkan jika *bar* disebelah kanan berarti sebuah *space* di sebelah kiri. Pengkodean disebelah kanan dinamakan kode *even parity* sebab unit ***black bar***-nya berjumlah genap. Sedangkan pengkodean disebelah kiri dinamakan kode *odd parity* sebab unit ***black bar***-nya berjumlah ganjil. Bilangan-bilangan yang dikodekan mempunyai perbedaan untuk tiap-tiap sisi *barcode*, sehingga *barcode* dapat dibaca (*scanned*) dari sebelah kiri maupun dari sebelah kanan.

Tabel berikut ini adalah pengkodean sisi kiri dan sisi kanan yang dipisahkan ke dalam tujuh unit.

Tabel 2.1 Pengkodean *barcode*



Penjelasan tabel pengkodean di atas adalah sebagai berikut :

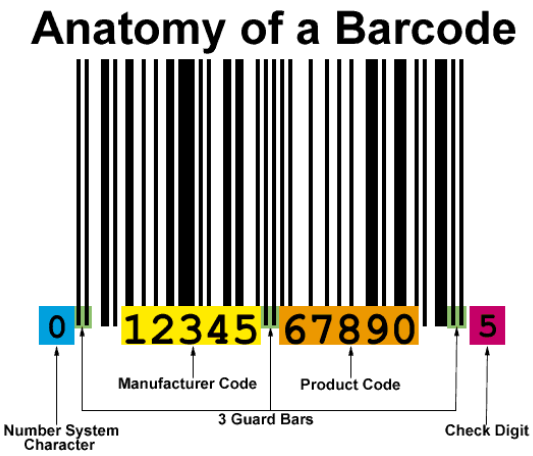
1. Seperti yang sebelumnya disebutkan bahwa bilangan-bilangan sebelah kiri merupakan kebalikan dari bilangan-bilangan disebelah kanan.

2. Setiap *barcode* memiliki empat buah "***mark***" (marka) yang berbeda. Sebuah marka dapat terdiri dari salah satu *black* (*bar*) atau *white* (*space*). Marka-marka tersebut lebarnya bermacam-macam, tetapi jumlahnya selalu empat. Contohnya, bilangan pengkodean yang berada di sebelah kiri pada bagian angka "0" yaitu 0001101 berarti terdiri dari 3 *space* (marka 1), 2 *bar* (marka 2), 1 *space* (marka 3), dan 1 *bar* (marka 4).
3. Pengkodean di sisi kiri selalu dimulai dengan sebuah *space* atau "0" dan berakhir dengan sebuah *bar* atau "1". Sedangkan untuk sisi sebelah kanan selalu dimulai dengan sebuah *bar* atau "1" dan berakhir dengan sebuah *space* atau "0".

   Untuk lebih jelasnya, lihat tabel *spesifikasi barcode* jenis UPC berikut ini :

Tabel 2.2 Spesifikasi *barcode* jenis UPC

	Sebelah kiri	Sebelah Kanan	Width
	(Odd/ganjil)	(Even/genap)	Pattern (mark)
0	0001101	1110010	3, 2, 1, 1
1	0011001	1100110	2, 2, 2, 1
2	0010011	1101100	2, 1, 2, 2
3	0111101	1000010	1, 4, 1, 1
4	0100011	1011100	1, 1, 3, 2
5	0110001	1001110	1, 2, 3, 1
6	0101111	1010000	1, 1, 1, 4
7	0111011	1000100	1, 3, 1, 2
8	0110111	1001000	1, 2, 1, 3
9	0001011	1110100	3, 1, 1, 2



Gambar 2.14 Anatomi *barcode*

Keterangan gambar *barcode* :

Komputer tidak membaca bilangan yang berada di bagian bawah *barcode*, tetapi bilangan tersebut dicetak agar orang dapat membaca *barcode* dengan mudah bila diperlukan.

1. *Number System Character*

   Angka ini merupakan sebuah sistem bilangan *barcode* UPC yang mengkarakteristikan jenis-jenis khusus pada *barcode*. Di dalam *barcode* UPC, *Number System Character* ini biasanya terletak disebelah kiri *barcode.*

   Kode-kode pada *Number System Character* adalah sebagai berikut :

   a. 0 - *Standard* UPC *number*.

   b. 1 - *Reserved.*

   c. 2 - *Random weight items like fruits, vegetables, and meats, etc.*

   d. 3 - *Pharmaceuticals.*

   e. 4 - *In-store code for retailers.*

   f. 5 - *Coupons.*

g. 6 - *Standard UPC number.*

h. 7 - *Standard UPC number.*

i. 8 - *Reserved.*

j. 9 - *Reserved.*

2. 3 *Guard Bars*

   Ada tiga *guard bars* yang ditempatkan di awal, tengah dan akhir pada *barcode*. *Guard bars* bagian awal dan akhir di-*encode*-kan sebagai "bar-space-bar" atau "101". *Guard bar* bagian tengah di-*encode*-kan sebagai "space-bar-space-bar-space" atau "01010".

   a. *Manufacturer Code* : kode perusahaan ini ada lima digit bilangan yang secara khusus menentukan manufaktur suatu produk. Kode perusahaan/manufaktur ini dilindungi dan ditetapkan oleh *Uniform Code Council* (UCC).

   b. *Product Code* : kode produk ini ada lima digit bilangan yang ditetapkan oleh perusahaan/manufaktur untuk setiap produk yang dihasilkannya. Untuk setiap produk yang berbeda dan setiap ukuran yang berbeda, akan memiliki kode produk yang unik.

3. *Check digit* : disebut sebagai digit "*self-check*". *Check digit* ini terletak di bagian luar sebelah kanan *barcode*. *Check digit* ini merupakan suatu " old-programmer's trick" untuk mengvalidasikan digit-digit lainnya (*number system character, manufacturer code, product code*) yang dibaca secara teliti.

2.10 *Wabcam*

Wabcam adalah berupa *hardware* seperti layaknya kamera digital yang terhubung dengan komputer agar menghasilkan *image* atau *video.*