BAB II
DASAR TEORI

2.1 Algoritma Genetika

Algoritma genetika merupakan evaluasi atau perkembangan dunia komputer dalam bidang kecerdasan buatan (*artificial intelligence*). Kemunculan algoritma genetika ini terinspirasi oleh teori Darwin (walaupun pada kenyataannya teori tersebut terbukti keliru) dan teori-teori dalam ilmu biologi, sehingga banyak istilah dan konsep biologi yang digunakan dalam algoritma genetika, karena sesuai dengan namanya, proses-proses yang terjadi dalam algoritma genetika sama dengan apa yang terjadi pada evaluasi biologi. [4]

Algoritma genetika adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi kompleks, yang sulit dilakukan oleh metode konvernsional.

2.1.1 Sejarah Singkat Algoritma Genetika

Algoritma genetika pertama kali dirintis oleh John Holland dari Universitas Michigan pada tahun 1960-an, algoritma genetika telah diaplikasikan secara luas pada berbagai bidang. Algoritma Genetika banyak digunakan untuk memecahkan masalah optimasi, walaupun pada kenyataannya juga memiliki kemampuan yang baik untuk masalah-masalah selain optimasi. John Holland menyatakan bahwa setiap masalah yang berbentuk adaptasi (alam maupun buatan) dapat diformulasikan dalam teknologi genetika. Algoritma genetika adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom.[5]

Lalu tujuh belas tahun kemudian, John Koza melakukan penelitian suatu program yang berkembang dengan menggunakan algoritma genetika program yang dikenal dengan sebutan metode "*genetic programming"* tersebut dibuat menggunakan LIPS (bahasa pemogramannya dapat dinyatakan dalam bentuk parse *tree* yaitu objek kerjanya pada algoritma genetika). Sampai sekarang

algoritma genetika terus digunakan untuk memecahkan permasalahan yang sulit dipecahkan dengan algoritma konvensional.

2.1.2 Aplikasi Algoritma Genetika

Sejak pertama kali dirintis oleh John Holland pada tahun 1960-an, algoritma genetika telah dipelajari, diteliti dan diaplikasikan secara luas pada berbagai bidang. Algoritma banyak digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter optimal**.** [3]

Keuntungan penggunaan algoritma genetika sangat jelas terlihat dari kemudahan implementasi dan kemampuannya untuk menentukan solusi seperti:

1. Ruang masalah sangat besar, kompleks, dan sulit dipahami.
2. Kurang atau bahkan tidak ada pengetahuan yang memandai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit.
3. Tidak tersedianya analisis matematika yang memadai.
4. Ketika metode-metode konvensional sudah tidak mampu menyelesaikan masalah yang dihadapi.
5. Solusi yang diharapkan tidak harus paling optimal, tetapi cukup bisa diterima.
6. Terdapat batasan waktu, misalnya dalam *real time systems* atau system waktu nyata.

Algoritma genetika telah banyak diaplikasikan untuk penyelesaian masalah dan pemodelan dalam bidang teknologi, bisnis dan entertainment seperti:

1. Optimasi

   Algoritma Genetika untuk optimasi *numeric* dan optimasi kombinatorial seperti *Traveling Salesman Problem* (TSP), perancangan *Intergrated Circuit* atau IC [LOU93], *job shop scheduling* [GOL91], optimasi video, dan suara.
2. Pemograman otomatis

   Algoritma genetika telah digunakan untuk melakukan proses evolusi terhadap program komputer untuk merancang struktur komputasional, seperti *cellular automatis* dan *sorting networks*.

3. *Machine learning*

   Algoritma genetika telah berhasil diaplikasikan untuk memprediksi struktur protein. Algoritma genetika juga berhasil diaplikasikan dalam perancangan *neural networks* (jaringan syaraf tiruan) untuk melakukan proses evolusi terhadap aturan-aturan pada *learning classifier systems* atau *symbolic prosuction systems*. Algoritma genetika juga digunakan untuk mengontrol robot.

4. Model Ekonomi

   Algoritma genetika telah digunakan untuk memodelkan proses-proses inovasi dan pembangunan *bidding strategies*.

5. Model Sistem Imunisasi

   Algoritma genetika telah berhasil digunakan untuk memodelkan berbagai aspek pada sistem imunisasi alamiah, termasuk *somatic mulation* selama kehidupan individu dan menentukan keluarga dengan gen ganda (*multi - gen families*) sepanjang waktu evolusi.

6. Model Ekologis

   Algoritma genetika telah berhasil digunakan untuk memodelkan fenomena ekologis seperti *host-parasite co-evolutions*, simbiosis dan aliran sumber daya dalam ekologi.

7. Interaksi antara Evolusi dan Belajar

   Algoritma genetika telah digunakan untuk mempelajari bagaimana proses belajar suatu individu bisa mempengaruhi proses evolusi suatu *species* dan sebaliknya.

2.1.3 Komponen Algoritma Genetika

Ada beberapa komponen algoritma genetika yang perlu diketahui sebelum pembuatan program diantaranya yaitu:

2.1.3.1 Teknik Pengkodean

Teknik pengkodean adalah bagaimana mengodekan gen dari kromosom, dimana gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variable. Agar dapat diproses melalui algoritma genetik, maka *alternative* solusi tersebut harus dikodekan terlebih dahulu kedalam bentuk kromosom. Masing-masing kromosom berisi sejumlah gen yang mengodekan informasi yang disimpan didalam kromosom.

Gen dapat direpresentasikan dalam bentuk : bit, bilangan real, daftar aturan, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

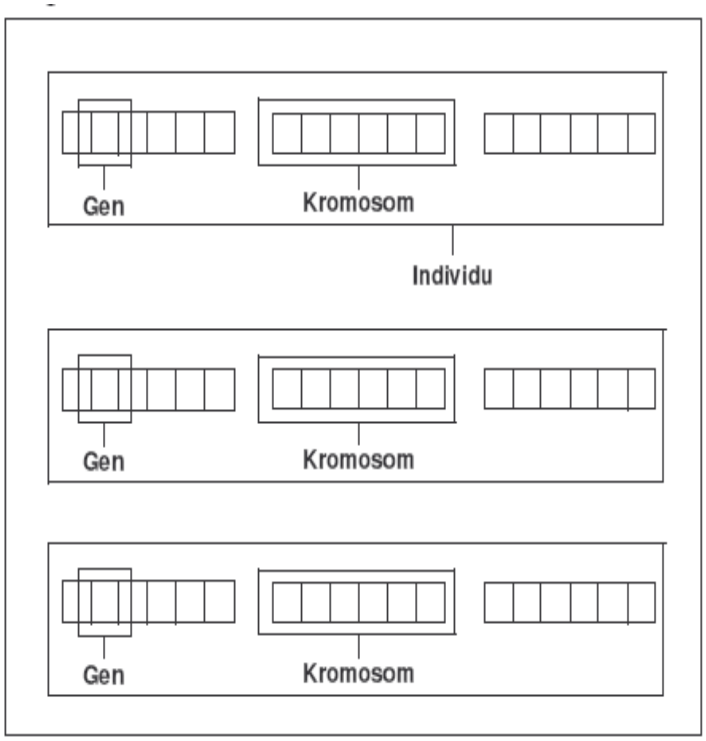
2.1.3.2 Pengertian Individu

Individu menyatakan salah satu solusi yang mungkin. Individu dapat dikatakan sama dengan kromosom, yang merupakan kumpulan gen. Beberapa definisi penting yang perlu diperhatikan dalam membangun penyelesain masalah menggunakan algoritma genetika, yakni:

1. *Genotype* (Gen), sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa berupa biner, float, interger maupun karakter, atau kombinatorial
2. Kromosom, gabungan gen-gen yang membentuk nilai tertentu.
3. *Allele*, merupakan nilai dari gen
4. Individu, menyatakan suatu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
5. Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evalusi.
6. Seleksi, merupakan proses untuk mendapatkan calon individu yang baik.

7. *Crossover*, merupakn proses pertukaran atau kawin silang gen-gen dari dua induk tertentu.

Populasi



Gambar 2.1 Individu dalam algoritma genetik

Hal-hal yang harus dilakukan dalam Algoritma genetika yaitu:

1. Mendefinisikan individu, dimana individu menyatakan salah satu solusi (penyelesaian) yang mungkin dari permasalahan yang diangkat.
2. Mendefinisikan nilai *fitness*, yang merupakan ukuran baik-tidaknya sebuah individu baik-tidaknya solusi yang didapat.
3. Menentukan proses pembangkitan populasi awal. Hal ini biasanya dilakukan dengan menggunakan pembangkitan acak seperti *ramdom-walk*.
4. Menentukan proses seleksi yang akan digunakan.
5. Menentukan proses perkawinan silang (*cross-over*) dan mutasi gen yang akan digunakan.

Hal penting yang harus diketahui dalam pemakaian Algoritma genetika:

1. Algoritma genetika adalah algoritma yang dikembangkan dari proses pencarian solusi menggunakan pencarian acak, ini terlihat pada proses pembangkitan populasi awal yang menyatakan sekumpulan solusi yang dipilih secara acak.
2. Berikut ini pencarian dilakukan berdasarkan proses teori genetika yang memperhatikan pemikiran bagaimana memperoleh individu yang lebih baik, sehingga dalam proses evaluasi dapat diharapkan diperoleh individu yang baik.

2.1.3.3 Membangkitkan Populasi Awal

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu secara acak atau melalui *procedure* tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal. [8]

Teknik dalam pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah sebagai berikut:

1. Seperti pada metode *random seach*, pencarian solusi dimulai dari suatu titik uji tertentu. Titik uji tersebut dianggap sebagai alternative solusi yang disebut sebagai populasi.
2. *Random Generator*

   Random generator adalah melibatkan pembangkitan bilangan random untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.
3. Pendekatan tertentu (memasukan nilai tertentu kedalam gen)

   Cara ini adalah dengan memasukan nilai tertentu kedalam gen dari populasi awal yang dibentuk.
4. Permutasi Gen

   Cara ini adalah penggunaan permutasi josephus dalam permasalahan kombinatorial seperti TSP.

2.1.3.4 Fungsi fitness

Suatu individu atau kromosom dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performasinya. Fungsi yang digunakan untuk mengukur nilai kecocokan atau derajat optimalitas suatu kromosom disebut dengan *fitness function.*

Nilai yang dihasilkan dari fungsi tersebut menandakan seberapa optimal solusi yang diperoleh. Nilai yang dihasilkan oleh fungsi fitness merepresentasikan seberapa banyak jumlah persyaratan yang dilanggar, sehingga dalam kasus penjadwalan perkuliahan semakin kecil jumlah pelanggaran yang dihasilkan maka solusi yang dihsilkan akan semakin baik.

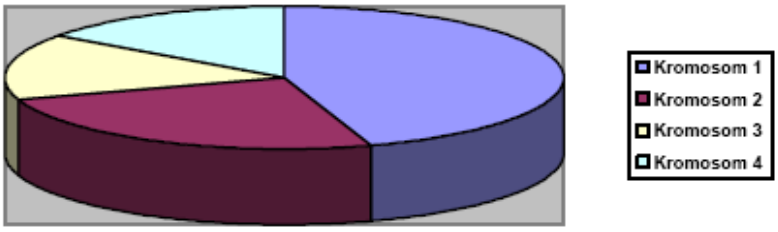
2.1.3.5 Seleksi

Setiap kromosom yang terdapat dalam populasi akan melalui proses seleksi untuk dipilih menjadi orang tua. Sesuai dengan teori Evolusi Darwin maka kromosom yang baik akan bertahan dan menghasilkan keturunan yang baru untuk generasi selanjutnya.[7]

Ada beberapa metode seleksi, yaitu:

1. Seleksi *Roulette Wheel*

   Model seleksi ini merupakan model yang paling besar variansinya. Munculnya individu superior sering terjadi pada model ini, sehingga perlu strategi lain menangani hal ini.

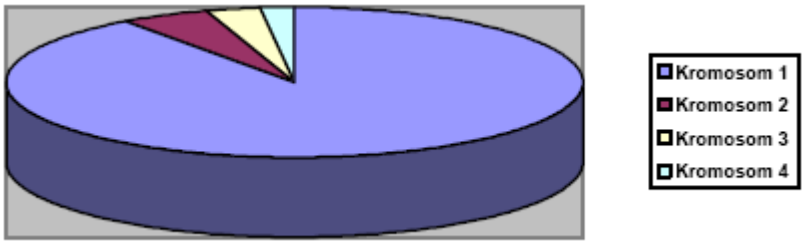


Gambar 2.2 *roulette wheel*

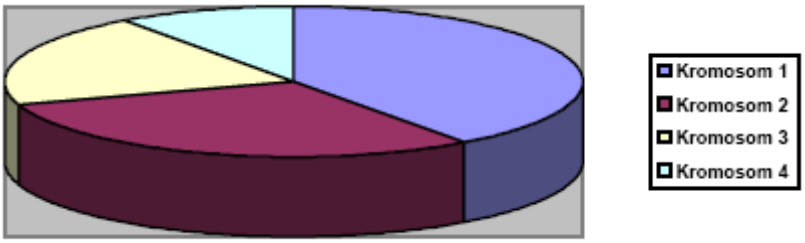
2. Seleksi Rangking

   Seleksi ini memperbaiki proses seleksi yang sebelumnya yaitu *roulette wheel* karena pada seleksi tersebut kemungkinan selain satu kromosom mempunyai nilai fitness yang mendominasi hingga 90% bisa terjadi.

Sehingga nilai fitness yang lain akan mempunyai kemungkinan yang sangat kecil untuk terpilih. Seleksi rangking dipakai untuk mengatasi masalah di atas, pertama-tama, diurutkan seluruh kromosom berdasarkan bagus-tidaknya solusi berdasarkan nilai *fitness*-nya. Setelah diurutkan, kromosom terburuk diberi nilai fitness baru sebesar 1, kromosom kedua terburuk diberi nilai *fitness* baru sebesar 2, dan seterusnya. Kromosom terbaik diberi nilai *fitness* baru sebesar n dimana n adalah banyak kromosom dalam suatu populasi.



Gambar 2.3 sebelum dirangking



Gambar 2.4 sesudah dirangking

3. Seleksi *Steady State*

   Metode ini tidak banyak digunakan dalam proses seleksi karena dilakukan dengan mempertahankan individu yang terbaik. Pada setiap generasi, akan dipilih beberapa kromosom-kromosom yang memiliki nilai fitness terburuk akan digantikan dengan *offspring* yang baru. Sehingga pada generasi selanjutnya akan terdapat beberapa populasi yang dipertahankan.

4. Seleksi Turnamen

   Merupakan metode seleksi lainnya yang didasari fenomena alamiah seperti turnamen antar individu dalam populasi. Dilakukan dengan memilih secara acak beberapa kromosom dari populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk.

5. Truncation Random

   Metode ini lebih mudah diterapkan jika dibandingkan dengan metode *Roulette Wheel*, pemilihan kromosom dilakukan secara acak tetapi tidak semua kromosom mendapatkan kesempatan tersebut, hanya kromosom terbaik saja yang berpeluang.

2.1.3.6 *Cross-Over* atau Kawin Silang

Proses kawin silang adalah salah satu operator penting dalam algoritma genetika, metode dan tipe *crossover* yang dilakukan tergantung dari *encoding* dan permasalahan yang diangkat. Ada beberapa cara yang bisa digunakan untuk melakukan *crossover* sesuai dengan encodingnya sebagai berikut: [5]

1. Binary encoding
   a. *Crossover* satu titik

      Memilih satu titik tertentu, selanjutnya nilai biner sampai titik *crossover*nya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai biner dari induk kedua.

      Contoh

      11001011 + 1101**1111** = **11001111**

   b. *Crossover* dua titik

      Memilih dua titik tertentu, lalu nilai biner sampai titik *crossover* pertama pada induk pertama digunakan, dilanjutkan dengan nilai biner dari titik sampai titik kedua dari induk kedua, kemudian sisanya dilanjutkan nilai biner dari titik kedua induk pertama lagi.

      Contoh:

      110010**11** + 11**0111**11 = **11011111**

   c. *Crossover uniform*

      Nilai biner yang digunakan dipilih secara random dari kedua induk.

      Contoh

      1**10**01011 + 11011**11** = **11011111**

   d. *Crossover* aritmatka

      Suatu operasi aritmatika digunakan untuk menghasilkan *offspring* yang baru.

Contoh:

11001011 + 11011111 = 11001001 (**AND**)

2. *Permutation encoding*

   Memilih satu titik tertentu, nilai *permutation* sampai titik *crossover*. Pada induk pertama digunakan lalu sisanya dilakukan *scan* terlebih dahulu, jika nilai permutasi pada induk kedua belum ada pada *offspring* nilai tersebut ditambahkan.

   Contoh:

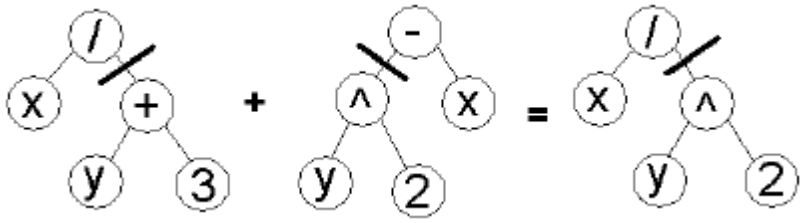
   (**12345**6789) + (4**53689**721) = **12345689**

3. *Value encoding*

   Semua metode *crossover* pada binary *crossover* bisa digunakan.

4. *Tree encoding*

   Memilih satu titik tertentu dari tiap induk, dan menggunakan *tree* dibawah titik pada induk pertama dan *tree* dibawah induk kedua.



Gambar 2.5 *Tree encoding.*

2.1.3.7 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Beberapa cara operasi mutasi yang diterapkan dalam algoritma genetika, antara lain: [8]

1. Mutasi dalam pengkodean biner.

   Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai bit pada posisi tertentu yang dipilih secara acak pada kromosom.

   Contoh.

   Kromosom sebelum mutasi : 1 0 0 1 0 **1** 1 1

   Kromosom sesudah mutasi : 1 0 0 1 0 **0** 1 1

2. Mutasi dalam pengkodean permutasi.

   Proses mutasi yang dilakukan pengkodean biner dengan mengubah langsung bit pada kromosom tidak dapat dilakukan pada pengkodean permutasi karena konsistensi urutan harus diperhatikan, salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi dari kromosom dan kemudian nilainya saling tukar.

   Contoh:

   Kromosom sebelum mutasi : 1 2 **3** 4 6 5 8 7 9

   Kromosom sesudah mutasi : 1 2 7 4 6 5 8 **3** 9

3. Mutasi dalam pengkodean nilai

   Proses mutasi dalam pengkodean nilai dapat dilakukan dengan berbagai cara, salah satunya yaitu dengan memilih sembarang posisi gen pada kromosom, nilai yang ada tersebut kemudian ditambah atau dikurangi dengan suatu nilai terkecil tertentu yang diambil secara acak.

   Contoh mutasi dalam pengkodean nilai riil dengan yang ditambahkan atau dikurangkan adalah 0,1

   Contoh:

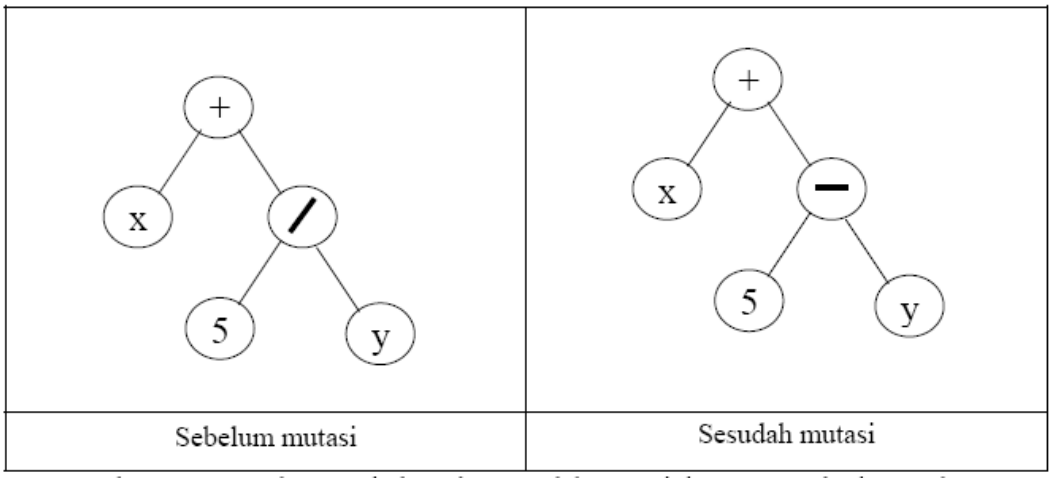
   Kromosom sebelum mutasi : 1,43 **1,09** 4,51 **9,11** 6,94

   Kromosom sesudah mutasi : 1,43 **1,19** 4,51 **9,01** 6,94

4. Mutasi dalam pengkodean pohon

   Mutasi dalam pengkodean pohon dapat dilakukan antara lain dengan cara mengubah operator (+, -, *, /) atau nilai yang terkandung dalam suatu verteks pohon yang dipilih. Atau dapat juga dilakukan dengan memilih dua verteks dari pohon dan saling mempertukarkan operator atau nilainya.

Contoh



Gambar 2.6 Mutasi dalam pengkodean pohon

2.1.4.8 Kondisi Selesai

Jika kondisi telah terpenuhi, maka algoritma genetika akan menghentikan proses pencariannya, tetapi jika belum terpenuhi maka algoritma genetika akan kembali ke evaluasi *fitness.*

2.2 Software yang mendukung

Software yang digunakan untuk membangun sistem dalam tugas akhir ini menggunakan Visual Basic 6.0 dan untuk membangun database menggunakan Microsoft Acces, penjelasan sebagai berikut:

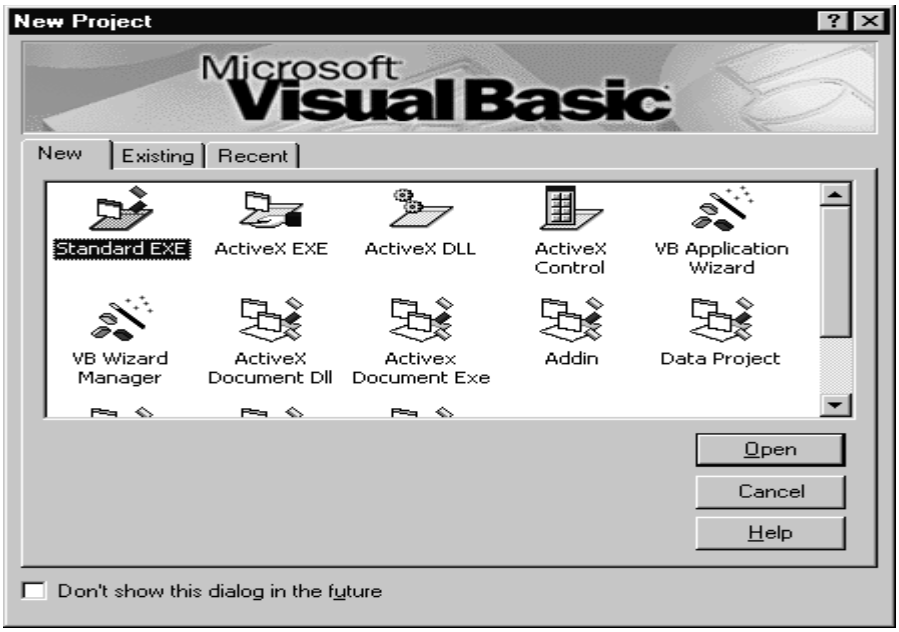
2.2.1 Visual Basic 6.0

Visual Basic adalah salah suatu developement tools untuk membangun aplikasi dalam lingkungan Windows. Dalam pengembangan aplikasi, Visual Basic menggunakan pendekatan Visual untuk merancang user interface dalam bentuk form, sedangkan untuk kodingnya menggunakan dialek bahasa Basic yang cenderung mudah dipelajari.[1]

IDE Visual Basic

Langkah awal dari belajar Visual Basic adalah mengenal IDE (Integrated Developement Environment) Visual Basic yang merupakan Lingkungan Pengembangan Terpadu bagi programmer dalam mengembangkan aplikasinya. Dengan menggunakan IDE programmer dapat membuat user interface, melakukan koding, melakukan testing dan debuging serta menkompilasi program menjadi executable. Penguasaan yang baik akan IDE akan sangat membantu programmer dalam mengefektifkan tugas-tugasnya sehingga dapat bekerja dengan efisien.

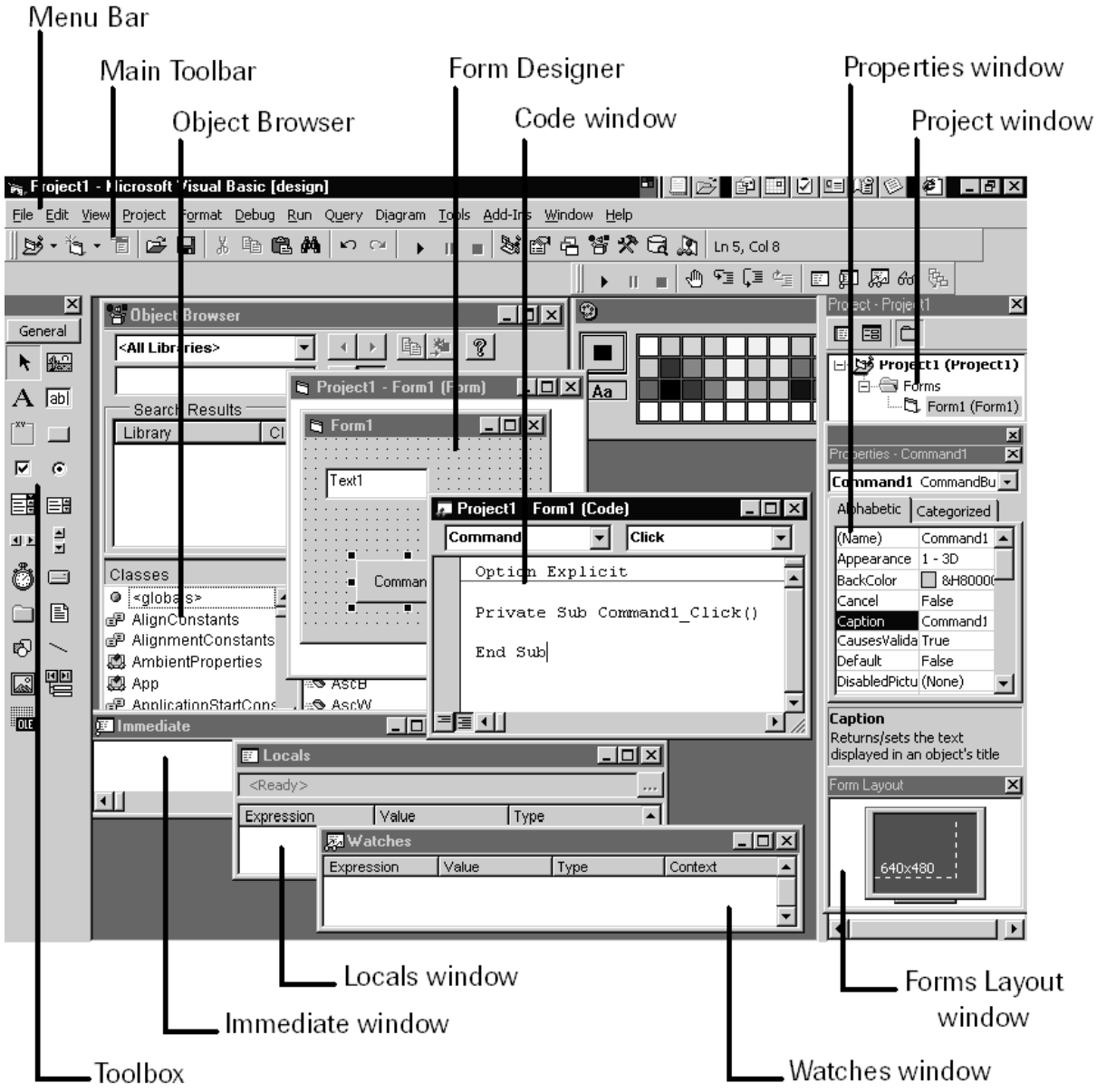
1. Memilih jenis Project



Gambar 2.7 Jenis Project

2. Jendela IDE

IDE Visual Basic 6 menggunakan model MDI (Multiple Document Interface). Berikut ini adalah gambar yang menunjukan bagian-bagian dan nama-nama jendela yang dapat tampil pada IDE Visual Basic.

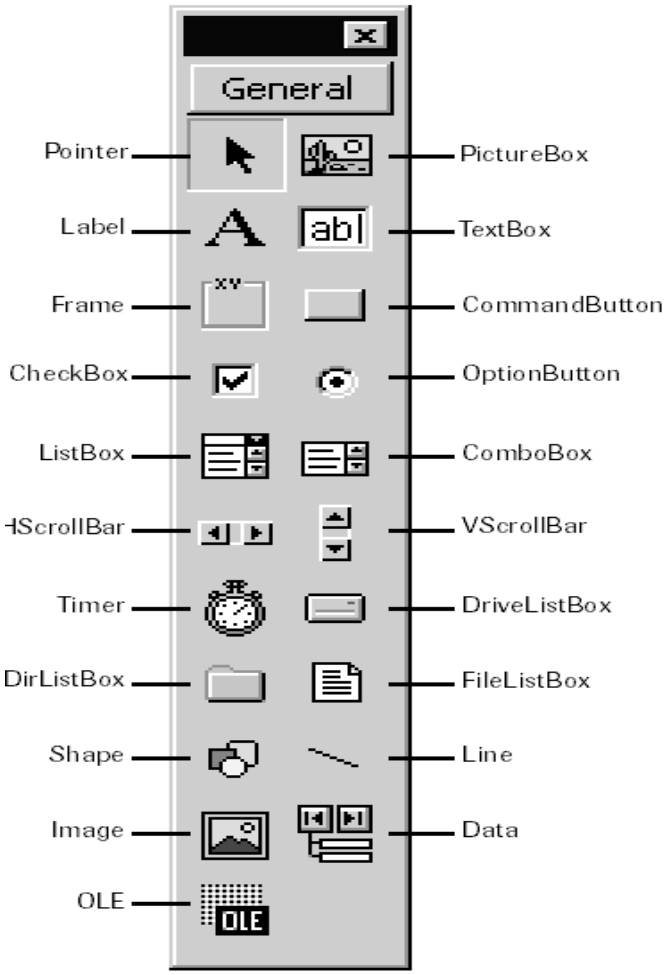


Gambar 2.8 Jendela IDE

Sebagaimana dengan proses belajar ini, kita akan fokus pada beberapa jendela yang penting terlebih dahulu sehingga konsentrasi tidak menjadi pecah, dan peserta belajar menjadi bingung. Adapun jendela-jendela yang perlu anda perhatikan adalah sebagai berikut :

- *Menu Bar*, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan project, membuka project, dll
- *Main Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.
- *Jendela Project*, jendela ini berisi gambaran dari semua modul yang terdapat dalam aplikasi anda. Anda dapat menggunakan icon Toggle Folders untuk menampilkan modul-modul dalam jendela tersebut secara di group atau berurut berdasarkan nama. Anda dapat menggunakan Ctrl+R untuk menampilkan jendela project, ataupun menggunakan icon Project Explorer.
- *Jendela Form Designer*, jendela ini merupakan tempat anda untuk merancang user interface dari aplikasi anda. Jadi jendela ini menyerupai kanvas bagi seorang pelukis.
- *Jendela Toolbox*, jendela ini berisi komponen-komponen yang dapat anda gunakan untuk mengembangkan user interface.
- *Jendela Code*, merupakan tempat bagi anda untuk menulis koding. Anda dapat menampilkan jendela ini dengan menggunakan kombinasi Shift-F7.
- *Jendela Properties*, merupakan daftar properti-properti object yang sedang terpilih. Sebagai contohnya anda dapat mengubah warna tulisan (foreground) dan warna latarbelakang (background). Anda dapat menggunakan F4 untuk menampilkan jendela properti.
- *Jendela Color Palette*, adalah fasilitas cepat untuk mengubah warna suatu object.
- *Jendela Form Layout*, akan menunjukan bagaimana form bersangkutan ditampilkan ketika runtime.

3. Toolbox



Gambar 2.9 Toolbox

Adapun secara garis besar fungsi dari masing-masing intrinsic kontrol tersebut adalah sebagai berikut :

- *Pointer* bukan merupakan suatu kontrol; gunakan icon ini ketika anda ingin memilih kontrol yang sudah berada pada form.
- *PictureBox* adalah kontrol yang digunakan untuk menampilkan image dengan format: BMP, DIB (bitmap), ICO (icon), CUR (cursor), WMF (metafile), EMF (enhanced metafile), GIF, dan JPEG.
- *Label* adalah kontrol yang digunakan untuk menampilkan teks yang tidak dapat diperbaiki oleh pemakai.
- *TextBox* adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.

- *Frame* adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.
- *CommandButton* merupakan kontrol hampir ditemukan pada setiap form, dan digunakan untuk membangkitkan event proses tertentu ketika pemakai melakukan klik padanya.
- *CheckBox* digunakan untuk pilihan yang isinya bernilai yes/no, true/false.
- *OptionButton* sering digunakan lebih dari satu sebagai pilihan terhadap beberapa option yang hanya dapat dipilih satu.
- *ListBox* mengandung sejumlah item, dan user dapat memilih lebih dari satu (bergantung pada property *MultiSelect*).
- *ComboBox* merupakan konbinasi dari TextBox dan suatu ListBox dimana pemasukkan data dapat dilakukan dengan pengetikkan maupun pemilihan.
- *HScrollBar* dan *VScrollBar* digunakan untuk membentuk scrollbar berdiri sendiri.
- *Timer* digunakan untuk proses background yang diaktifkan berdasarkan interval waktu tertentu. Merupakan kontrol non-visual.
- *DriveListBox*, *DirListBox*, dan *FileListBox* sering digunakan untuk membentuk dialog box yang berkaitan dengan file.
- *Shape* dan *Line* digunakan untuk menampilkan bentuk seperti garis, persegi, bulatan, oval.
- *Image* berfungsi menyerupai image box, tetapi tidak dapat digunakan sebagai kontainer bagi kontrol lainnya. Sesuatu yang perlu diketahui bahwa kontrol image menggunakan resource yang lebih kecil dibandingkan dengan PictureBox
- *Data* digunakan untuk *data binding*
- *OLE* dapat digunakan sebagai tempat bagi program eksternal seperti Microsoft Excel, Word, dll.

2.2.2 Microsoft Acces

Microsoft Access adalah suatu program *Microsoft office* pengolahan database. Para pengguna *microsoft access* umumnya hanya menggunakan program hanya untuk membuat tabel database. Padahal, didalam *microsoft access* tersedia fasilitas yang bias digunakan untuk membuat laporan dalam bentuk tabel, *query*, maupun *report* database secara maksimal.[2]

Jenis-jenis *Query*:

1. *Select queries*.

   Query ini menampilkan ekstrak data atau seluruh data. Data yang ditampilkan bias dipilih sesuai kriteria tertentu. Contoh yang ditampilkan hanya *client city* dibandung. Selain itu, bias menampilkan *field* hasil perhitungan dengan formula tertentu.

2. *Action queries.*

   Query ini digunakan untuk melakukan *delete, update, insert* data, dan *create* tabel baru.

3. *Cross tab queries.*

   Query ini digunakan untuk mengkakulasi data untuk keperluan analisis. *Crosstab queries* menghitung sebuah *count, average, sum,* atau tipe total lainnya yang dikelompokan kedalam dua jenis informasi.