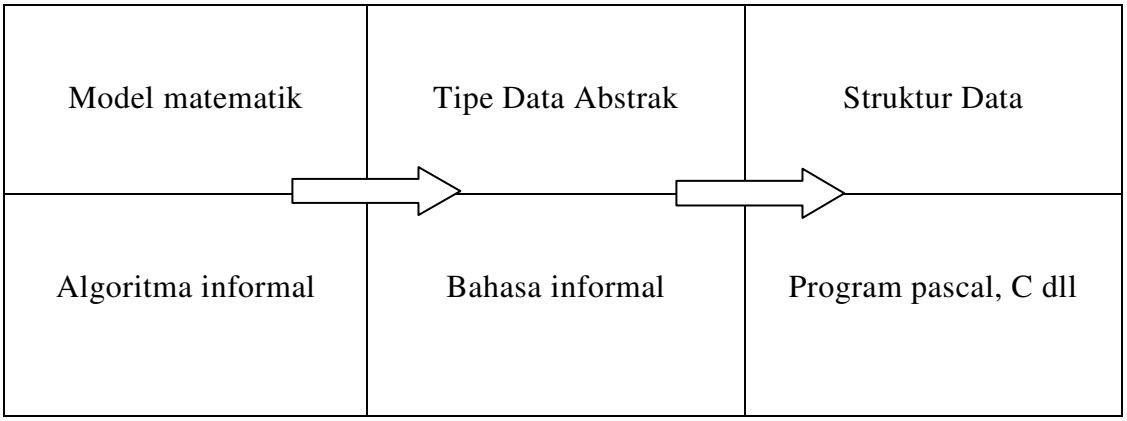
BAB II

LANDASAN TEORI

2.1. Tipe Data Abstrak (TDA)

"Tipe data sebuah variabel adalah kumpulan nilai yang dapat dimuat oleh variabel ini. Misalnya sebuah tipe boolean hanya bernilai TRUE atau FALSE, tidak boleh nilai yang lain. TDA adalah suatu model matematika, disertai sekumpulan operasi terhadap model itu" (Slamet, 1990:14). Untuk merepresentasikan suatu model matematika dari suatu TDA, digunakan struktur data yang berisi sekumpulan variabel, yang bisa terdiri atas beberapa tipe data, dan mempunyai bermacam-macam jenis dan cara relasi antara setiap variabel. Contoh sebuah TDA yang sangat sederhana adalah himpunan bilangan bulat {-3,4,0,5,…} dan operasi yang bisa dilakukan terhadap himpunan ini {gabungan, irisan dan lain-lain}. Dalam sebuah TDA, operasi-operasi dapat mengambil operand bukan hanya elemen TDA itu, tapi juga TDA yang lainnya. Demikian juga hasil operasinya bukan hanya merupakan elemen TDA itu, tapi juga TDA yang lain. Implementasi dari model matematik yang digunakan untuk menyelesaikan persoalan ke dalam program yang akan ditulis dilakukan dengan menterjemahkan model ke dalam struktur data abstrak (TDA) atau *abstract data type* yang selanjutnya dituliskan sebagai suatu struktur data yang sebenarnya dituliskan dalam bahasa pemrograman yang akan digunakan, seperti Pascal, C atau yang lainnya. Proses penulisan progam ini dapat digambarkan dalam bagan berikut :



2.2. Tumpukan (Stack)

Tumpukan sama dengan array, tetapi tumpukan bersifat dinamis sedangkan array bersifat statis. Dalam array, pengaksesan data (penambahan / penghapusan data) dapat dilakukan pada sembarang elemen (dari depan, tengah maupun belakang). Tetapi pada tumpukan pengaksesan data hanya dapat dilakukan pada satu ujung saja. "Secara sederhana, tumpukan bisa diartikan sebagai suatu kumpulan data yang seolah-olah ada data yang diletakkan diatas data yang lain" (Santosa, 1992:71). Kita bisa menambahi (menyisipkan) data, dan mengambil (menghapus) data lewat ujung yang sama, yang disebut ujung atas tumpukan (top of stack).

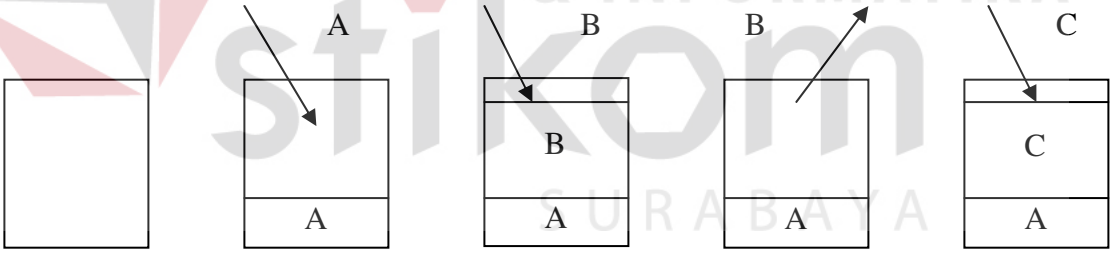
Untuk menjelaskan pengertian diatas kita ambil contoh sebagai berikut, misalnya kita mempunyai dua buah kotak yang kita tumpuk, sehingga kotak kita letakkan diatas kotak yang lain. Jika kemudian tumpukan dua buah kotak itu kita tambah dengan kotak ketiga, keempat, dan seterusnya, maka akan kita peroleh sebuah tumpukan kotak, yang terdiri dari N kotak.

"Ada dua operasi dasar yang bisa kita lakukan pada sebuah tumpukan, yaitu operasi menyisipkan data, atau mempush data, dan operasi menghapus data atau mempop data" (Santosa, 1992:74). Dengan demikian jika akan dilakukan penghapusan data, data yang akan dihapus justru data yang paling baru disisipkan.

"Operasi-operasi dasar yang dapat dilakukan terhadap stack adalah :

MAKENULL(S)	kosongkan stack S
TOP(S)	ambil elemen top stack
POP(S)	hapus elemen pada top stack
PUSH(x,S)	sisipkan elemen x pada top stack
EMPTY(S)	cek kekosongan stack pada S" (Slamet, 1990:30)

Proses dalam stack dapat dilihat pada gambar 2.2.



Gambar 2.1 Proses Push dan Pop pada stack

2.3. Array

"Array adalah tipe terstruktur yang mempunyai komponen dalam jumlah yang tetap dan setiap komponen mempunyai tipe data yang sama. Posisi masing-masing komponen dalam array dinyatakan sebagai nomor index" (Santosa, 1992:18). Array terdiri atas serangkaian sel bertipe data tertentu, nilai suatu sel dapat diambil dengan menyebutkan indeks sel itu, yaitu letak sel didalam array.

Bentuk umum dari deklarasi tipe array adalah :

type stack = array [tipe_index] of tstack;

dengan stack : nama tipe data

tipe_index : tipe data untuk nomor index

tstack : tipe data komponen

Parameter tipe_index menentukan banyaknya komponen array tersebut. Parameter ini boleh berupa sembarang tipe ordinal kecuali **longint** dan subjangkauan dari **longint.**

Sebagai, contoh, deklarasi :

Type stack = array[1..100] of integer;

Menunjukkan bahwa Stack adalah tipe data yang berupa array yang komponennya bertipe integer dan banyaknya 100 buah.

2.4. Infix

Ungkapan aritmatika terdiri atas operand dan operator dan umumnya ungkapan tersebut dituliskan dalam bentuk notasi infix, dimana operator ditulis diantara operand, contoh :

A + B * C / (D + E)

Ungkapan ini terdiri atas 5 operand yaitu A,B,C,D,E dan operatornya adalah * + pembatasnya adalah karakter blank. Operand dapat berupa variabel atau konstanta, operatornya meliputi operator dasar * / + - ^ (kali, bagi, tambah, kurang, dan pangkat), juga operator-operator tambahan plus dan minus unari, mod, div, rem, ceil, dan floor.

Masalahnya adalah bagaimana mengartikan ungkapan ini, yaitu bagaimana urutan pelaksanaan operator terhadap operand-operand. Untuk itu dibuat suatu hirarki umum :

operator	prioritas
^ (pangkat)	5
plus dan minus unari	4
* / div mod rem	3
+ -	2
(…)	1

Dengan demikian cara evaluasi ungkapan diatas adalah pasangan operand yang akan dievaluasi lebih dulu adalah pasangan operand dibatasi oleh operator yang paling tinggi prioritasnya. Sebagai ilustrasi evaluasi ungkapan diatas akan menjadi :

A + B * C / (D + E)

Hitung B * C

Hitung D + E

Hitung (B * C) / (D + E)

Hitung A + ((B * C) / (D + E))

Hasil

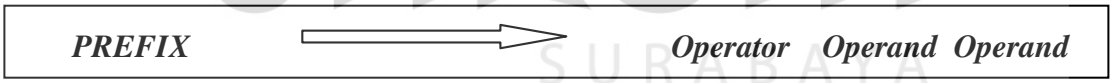
2.5. Prefix

"Seorang ahli matematika Jan Lukasiewich mengembangkan satu cara penulisan ungkapan numeris yang selanjutnya disebut *Polish Notation atau notasi Prefix* yang artinya bahwa operator ditulis sebelum kedua operand yang akan disajikan" (Insap Santosa, 1992:81) Dari contoh dalam notasi infix maka bisa dibuat ke dalam bentuk notasi prefix, yaitu :

A / * H + B1 B2 2

- A B

- + A B C



2.6. Postfix

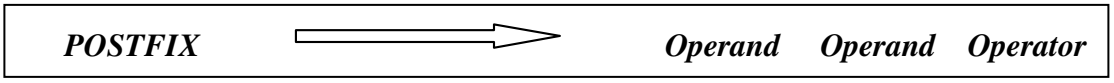
Pada notasi postfix, operator dituliskan sesudah atau dibelakang operand-operand yang akan digunakan. Notasi postfix biasa disebut juga dengan *Reverse Polish notation atau RPN*. Keuntungan utama pada notasi postfix adalah adanya kepastian operand-operand yang harus digunakan oleh suatu operator, operator yang harus digunakan adalah kedua operand yang ada didepan operator. Dengan

demikian pada notasi postfix tidak diperlukan adanya tanda kurung untuk memperjelas arti ungkapan. Contoh :

A H B1 B2 + * 2 / =

A B + C –

A B + C D - *



2.7. Konversi Notasi Infix ke Postfix

Salah satu kegunaan dari stack adalah untuk melakukan konversi dari suatu ungkapan aritmatik dalam bentuk infix ke dalam ungkapan dalam bentuk postfix. Dengan notasi infix, komputer akan sukar mengevaluasi ungkapan tersebut. Untuk itu perlu digunakan notasi postfix untuk menyatakan ungkapan yang sama. Dalam hal ini operator ditulis sesudah operand dan tidak memerlukan tanda kurung untuk memperjelas arti ungkapan. Contoh :

Notasi Infix	*Notasi Postfix*
A + B	A B +
A + B – C	A B + C –
(A + B) * (C – D)	A B + C D - *
A / B ^ C + D	A B C ^ / D +

2.8. Konversi Notasi Infix ke Prefix

Secara sederhana, proses konversi dari infix menjadi prefix dijelaskan sebagai berikut, misalkan ungkapan yang akan dikonversikan adalah :

(A + B) * (C – D)

Dengan menggunakan tanda kurung bantuan, ungkapan diatas diubah menjadi [+ A B] * [- C D]

Jika [- A B] dimisalkan P, dan [- C D] dimisalkan Q, maka ungkapan diatas bisa ditulis sebagai :

P * Q

Selanjutnya, notasi infix diatas diubah menjadi notasi prefix :

* P Q

Dengan mengembalikan P dan Q pada notasinya semula dan menghapus tanda kurung bantuan, diperoleh notasi prefix dari persamaan (A+B) * (C–D), yaitu :

* + A B – C D

Dari contoh-contoh diatas bahwa dalam penulisan ungkapan, bahkan yang rumit sekalipun, tidak pernah digunakan tanda kurung untuk pengelompokan. Dalam hal ini urutan penulisan operator akan menentukan operasi mana yang harus dikerjakan lebih dahulu.

2.9. Konversi Notasi Prefix ke Postfix

Untuk mengkonversi notasi prefix ke postfix perlu diperhatikan beberapa pengetahuan dasar sebagai berikut :

<Op> <Pre1> <Pre2>

menjadi

<Post1> <Post2> <Op>

dan

<Pre1> menjadi <Post1>

<Pre2> menjadi <Post2>

Contoh :

Conv (+/AB-CD) ⇒ Conv (/AB) Conv (-CD) +

Conv (A) Conv(B) / Conv (C) Conv (D) - +

A B / C D - +

A B / C D - +

2.10. Konversi Notasi Postfix ke Prefix

Sama halnya dengan konversi notasi prefix ke postfix, maka dalam konversi notasi postfix ke prefix pun mengikuti beberapa pengetahuan dasar sebagai berikut :

<Post1> <Post2> <Op>

menjadi

<Op> <Pre1> <Pre2>

dan

<Post1> menjadi <Pre1>

<Post2> menjadi <Pre2>

Contoh :

Conv (AB/CD-+) ⇒ + Conv (AB/) Conv (CD-)

+ / Conv (A) Conv(B) - Conv (C) Conv (D)

+ / A B - Conv (C) Conv (D)

+ / A B - C D

2.11. Konversi Notasi Prefix ke Infix

Secara sederhana proses konversi notasi prefix ke infix bisa dijelaskan melalui aturan dasar sebagai berikut :

<Op> <Pre1> <Pre2>

menjadi

<Inf1> <Op> <Inf2>

dan

<Pre1> menjadi <Inf1>

<Pre2> menjadi <Inf2>

Contoh :

Conv (-+ABC) ⇒ Conv (+AB) - Conv (C)

Conv (A) + Conv(B) - Conv (C)

A + B - Conv (C)

A + B - C

2.12. Konversi Notasi Postfix ke Infix

Dalam mengkonversi notasi postfix menjadi infix perlu diperhatikan beberapa pengetahuan dasar sebagai berikut :

<Post1> <Post2> <Op>

menjadi

<Inf1> <OP> <Inf2>

dan

<Post1> menjadi <Inf1>

<Post2> menjadi <Inf2>

Contoh :

Conv (AB*C+) ⇒ Conv (AB*) + Conv (C)

Conv (A) * Conv(B) + Conv (C)

A * B + C

A * B + C

2.13. Delphi 6.0.

2.13.1 Mengenal Delphi

Delphi for Windows merupakan perangkat pengembang yang begitu memanjakan pemakainya. Demikian banyak kemudahan yang disediakan untuk menyusun program aplikasi. Dengan delphi, seorang awam dapat memasuki dunia pemrograman yang sebelumnya dianggap terlalu sukar untuk dijamah. Selain

kemudahan yang melimpah, Delphi juga menjanjikan waktu penyusunan aplikasi tercepat didunia saat ini.

Delphi adalah perangkat lunak untuk menyusun program aplikasi yang berdasarkan pada bahasa pemrograman Pascal dan bekerja dalam lingkungan sistem operasi Windows. Dengan Delphi, kita akan merasakan begitu mudahnya menyusun program aplikasi, karena Delphi menggunakan komponen-komponen yang akan menghemat penulisan program.

Dengan Delphi pengguna tidak perlu lagi repot membuat window, kotak dialog maupun perangkat kontrol lainnya seperti tombol perintah (command button), menu pulldown, kotak kombo (combo box) dan sebagainya. Karena semua komponen tersebut sudah disediakan oleh Delphi. Kita tinggal mengambilnya dengan sangat mudah : klik mouse (tidak perlu lagi menulis kode yang rumit untuk menggambarnya)

Selain kemudahan, kelebihan Delphi adalah dalam hal kecepatan proses kompilasi program. Sampai saat ini belum ada perangkat lunak bidang bahasa pemrograman yang mampu menyaingi kecepatannya. Dapat disebutkan pula sebagai keunggulan Delphi yaitu ukuran file *.exe (exe=excecution, yaitu file yang mampu berjalan sendiri diluar software pembangunnya) yang dihasilkannya relatif sangat kecil sehingga tidak menggunakan ruang disk yang besar. Selain itu, fil-file *.dll (dll=dynamic link library, yaitu file program bantu pada file execution) yang dihasilkannyapun mampu berdiri sendiri tanpa harus melalui program utamanya. Masih banyak kekuatan Delphi yang sepertinya tidak akan

habis dibahas pada kesempatan ini, kita rasakan sendiri saja bedanya, seperti pemakaian jalur 32 bit untuk pengaksesan data dan lain-lain.

2.13.2 Visual Component Library

Visual Component Library (VCL) berisi kumpulan objek yang ditulis dalam Object Pascal dari Delphi untuk dipakai pada form Delphi. Delphi menyediakan VCL yang dapat diakses dari Component Palette.

Component Palette merupakan bagian dari toolbar. Ada dua jenis komponen yang terdapat dalam component palette, yaitu visible (tampak) dan invisible (tidak tampak).

- Komponen visible (tampak) adalah komponen-komponen yang diletakkan dalam form atau ditempatkan dalam program oleh pemakai.

  Misalnya Label, Edit Button, Memo dan lain-lain. Komponen-komponen tersebut diletakkan pada form, dan keberadaannya akan ditampilkan dilembar kerja saat aplikasi dijalankan.

- Komponen invisible (tidak tampak) adalah komponen yang memberikan kontrol khusus atau sebagai antarmuka bagi pemrogram.

  Misalnya komponen Timer, Open Dialog, MainMenu dan lain-lain. Komponen-komponen tersebut diletakkan pada form yang keberadaannya tidak akan ditampilkan di lembar kerja form saat aplikasi dijalankan.

2.13.3 Event

Event meliputi syntax (bahasa pemrograman), parameter yang terkait, tipe data, nilai default, dan apakah parameter digunakan untuk input atau output. Dalam Delphi, hampir semua kode yang ditulis dapat dieksekusi baik secara langsung maupun tidak langsung dan bahkan untuk dieksekusi oleh event.

Event merupakan suatu jenis properti khusus yang akan dijalankan sebagai suatu kejadian, yang sering disebut aksi. Kode yang direspon secara langsung oleh sebuah event sering disebut dengan event handler yang merupakan suatu procedure Object Pascal.

2.14. INTERAKSI MANUSIA DAN KOMPUTER

Tujuan utama disusunnya berbagai cara interaksi manusia dan komputer pada dasarnya adalah untuk memudahkan manusia dalam mengoperasikan komputer dan mendapatkan berbagai umpan balik yang ia perlukan selama ia bekerja pada sebuah sistem komputer. Dengan kata lain, para perancang antarmuka manusia dan komputer berharap agar sistem komputer yang ia rancang dapat mempunyai sifat yang akrab dan ramah dengan penggunanya.

Sistem komputer terdiri atas 3 aspek yaitu perangkat keras (hardware), perangkat lunak (software) dan manusia (brainware), yang saling bekerja sama. Kerja sama tersebut ditunjukkan dalam kerja sama antara komputer dengan manusia. Komputer yang terdiri perangkat keras (hardware) dan perangkat lunak (software) digunakan oleh manusia dalam menghasilkan sesuatu sesuai dengan keinginan manusia. Yang dimaksud dengan interaksi manusia dan komputer adalah terjadinya

komunikasi antara pemakai (manusia) dengan komputer untuk saling bertukar informasi seperti layaknya percakapan orang dengan komputer melalui suatu masukan dan keluaran.

Untuk mengerti hubungan pemakai dengan komputer secara baik adalah dengan membaginya ke dalam 9 minimal :

1. Pemakai komputer

   Dalam membuat suatu interaksi harus memperhatikan siapa yang akan menggunakan system tersebut, agar nantinya pemakai tidak kebingungan dalam menggunakan system tersebut.

2. Alat input

   Alat input yang digunakan harus mudah dipakai oleh user sehingga tidak menemui kesulitan dalam penggunaannya.

3. Bahasa input

   Bahasa input yang digunakan harus mudah dimengerti oleh user

4. Rancangan dialog

   Rancangan dialog dibuat untuk memudahkan user dalam mengakomodasikan keinginannya.

5. Alat output

   Untuk dapat melihat hasil atau informasi yang dikeluarkan oleh system

6. Pemandu user

   Adanya suatu panduan yang dibuat untuk memudahkan user dalam menggunakan system yang diterapkan.

7. Pesan yang nampak dalam komputer

   Apabila melakukan kesalahan, ada suatu pesan kepada user yang akan sangat membantu, sebab kesalahan tersebut akan segera diperbaiki oleh user.

8. Rancangan output

   Merupakan hal yang sangat perlu diperhatikan karena akan berhubungan langsung dengan pandangan user. Apabila rancangan layar tidak sesuai maka ketertarikan user pada aplikasi menjadi kurang.

9. Waktu respon komputer

   Dalam penggunaan system, user biasanya membutuhkan kecepatan akses. Oleh karena itu yang perlu diperhatikan juga adalah bagaimana sistem itu dibuat agar user dapat mengaksesnya dengan cepat.