BAB II

LANDASAN TEORI

2.1 Game Komputer

Mendefinisikan apakah yang dimaksud dengan game, tidak cukup dengan hanya melihat kamus bahasa. Terdapat banyak makna dalam kata *'game'*. Yang jelas *game* secara naluri adalah merupakan bagian dari kehidupan manusia. Makna sekilas dari *game* memberikan pengertian bahwa *game* merupakan suatu aktifitas yang tidak dilakukan dengan sungguh-sungguh. Untuk mengetahui apa yang sesungguhnya disebut dengan *game*, maka paling tidak kita dapat memahaminya dari adanya sejumlah pengertian *game* yang biasa kita alami dalam kehidupan.

Terdapat 5 kategori istilah *game*, yaitu :

1. *Board Games* (Permainan Papan) *Game* pada katagori ini membutuhkan suatu papan yang terbagi dalam sektor-sektor tertentu (dengan garis-garis) dan didalamnya terdapat sejumlah alat main yang dapat digerakkan. Termasuk dalam katagori ini adalah catur.Dua buah pemain akan berhadapan dan saling mengadu strategi sesuai dengan aturan untuk mencapai daerah lawan atau mempertahankan daerahnya sendiri, mengalahkan bidak musuh, mengumpulkan sesuatu. Pemain pada *board games* ini akan berusaha menganalisis hubungan-hubungan geometri yang ada pada papan dan bidak.
2. *Card Games* (Permainan Kartu) *Games* ini akan memanfaatkan simbol dari 52 kartu yang terbagi dalam dua faktor : *suit* (4 nilai) dan *rank* (13 nilai). Permainan akan dilakukan sekitar bagaimana membuat kombinasi dari 52

kartu tersebut. Sejumlah ketentuan dibuat untuk mengatur bagaimana cara-cara untuk membuat kombinasi tersebut. Permainan kartu *bridge/truf* termasuk kelompok games ini.

3. *Athletic Games* (Permainan Atletik) Permainan games jenis ini lebih cenderung pada penggunaan fisik daripada mental. Aturan *game* dibuat dengan keharusan pemain untuk melakukan sejumlah aksi tertentu. Hal yang terkait dengan kekuatan badan, kecepatan, ketepatan dan kerjasama menjadi bagian utama dari *game* atletik. Dalam hal ini harus dibedakan antara *game* dengan kompetisi. Kompetisi tidak mengharuskan adanya kerjasama dan pemain bekerja secara individu. Dua orang yang beradu lari bukanlah termasuk *game* tetapi kompetisi. Perbedaan utama antara *game* dan kompetisi adalah dalam hal interaksi diantara peserta. Pada kompetisi tidak terjadi interaksi diantara pemain, kompetisi yang membolehkan adanya interaksi diantara pemainnya termasuk *game*.
4. *Children Games* (Permainan Anak) Aktifitas seperti berlari, sembunyi, melempar dan menangkap adalah menjadi ciri utama *game* anak-anak. Umumnya *game* ini menekankan pada aktifitas kelompok sebagai latihan untuk berkehidupan sosial. Walaupun dalam *game* ini terdapat juga upaya untuk saling mengalahkan secara mental atau fisik namun tujuan utamanya bukanlah untuk meraih kemenangan satu diatas yang lain tetapi sebagai ilustrasi kerjasama dalam kehidupan manusia. Penggunaan sejumlah alat untuk membantu ilustrasi dapat meningkatkan improvisasi game dan meningkatkan keterlibatan yang lebih baik dari pemain.

5. *Computer Games* (Permainan Komputer) *Game* ini dimainkan lewat bantuan alat komputer. Terdapat 5 alat yang dapat dikategorikan sebagai komputer, yaitu :

   a. *Expensive dedicated machine,* mesin yang dioperasikan dengan koin untuk memainkankannya.

   b. *Inexpensive dedicated machine,* disebut juga dengan hand held machine. Alat game watch termasuk dalam katagori ini.

   c. *Multiprogram home,* mesin seperti Atari,Nintendo termasuk dalam kelompok komputer ini.

   d. *Personal computer*

   e. *Mainframe computer*

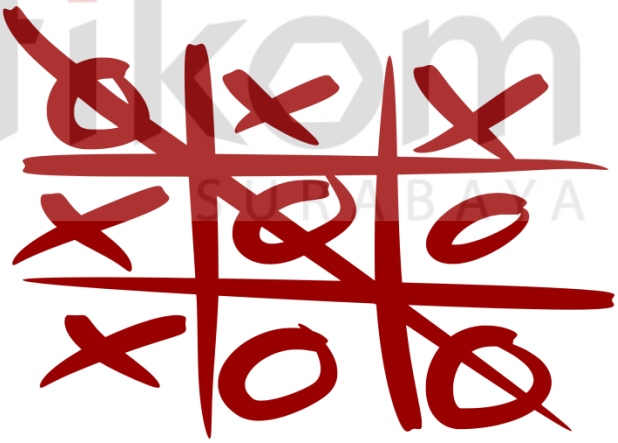
   Computer game berbeda dengan jenis *game* yang lain karena tidak ada pergerakan secara fisik atau interaksi langsung dengan object kecuali lewat perantaraan komputer. *Software* yang dibuat harus dapat menangkap reaksi yang cepat dari interaksi yang dihasilkan dengan pemain. Karena itu *software* untuk computer games harus bersifat real time. Kompleksitas *game* adalah bergantung dari kemampuan merepresentasikan aturan dan lingkungan game dalam program yang dibuat. Diantara sekian banyak definisi *game,* maka definisi yang umum untuk *computer game* adalah :

   Game Komputer adalah sebuah program software dimana satu atau lebih pemain berusaha untuk membuat keputusan lewat kontrol terhadap object dan resource guna memenuhi satu tujuan tertentu. (Sumber : prayudi,2008)

2.2 Tic Tac Toe

Tic Tac Toe adalah permainan untuk dua pemain dimana masing-masing pemain mengambil giliran dengan menandai ruang pada kotak 3x3. Biasanya ruang tersebut ditandai dengan simbol X dan O. X merupakan pemain satu dan O sebagai pemain dua atau sebaliknya. Ada tiga jenis dari hasil hasil permainan *Tic Tac Toe* yaitu menang, kalah atau imbang. Dinyatakan menang dalam permainan *Tic Tac Toe* apabila salah satu pemain berhasil mengisi simbol pada tiga ruang di horisontal, vertikal, atau diagonal. Beberapa aturan yang sudah di tentukan dalam permainan *Tic Tac Toe* , berikut aturan tersebut :

a) Pemain dengan simbol X mempunyai satu lebih banyak atau sama dengan jumlah dari simbol O dengan asumsi pemain pertama dimainkan pertama
b) Kedua pemain tidak ada yang bergerak jika sudah ada pemain yang menang. (Sriram,Vijayarr, Raghuras, Xyuan,2009)



Gambar 2.1 Game Tic Tac Toe

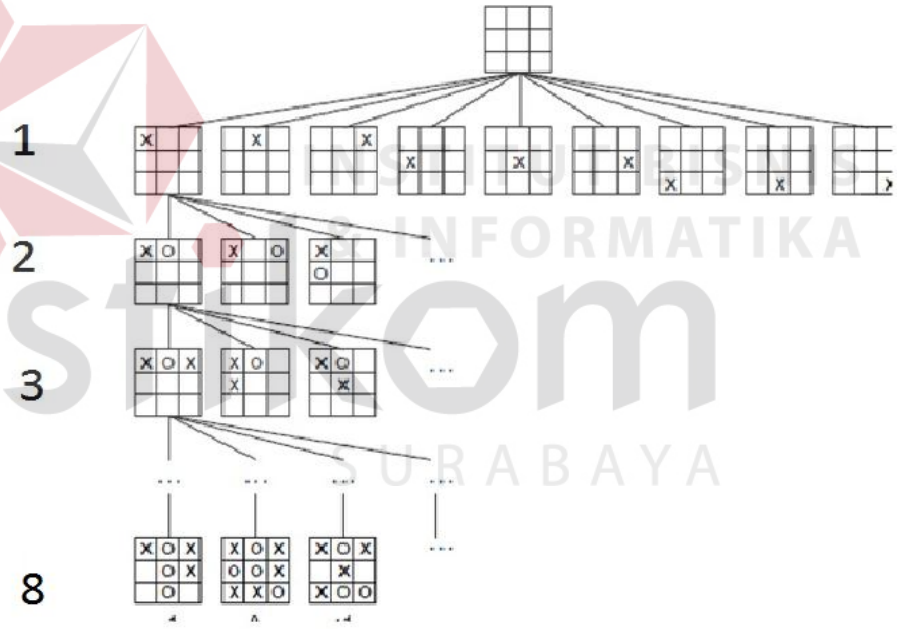
(Sumber: https://en.wikipedia.org/wiki/Tic-tac-toe**)**

Pada gambar di atas pemain dengan simbol O dinyatakan menang karena telah berhasil mengisi simbol O pada tiga ruang pada posisi diagonal.

2.3 Game Tree

Sebuah model permainan berbentuk pohon yang dimainkan oleh 2 (dua) pemain : manusia dan komputer. Para pemain saling bergantian melangkah . kedua pemain sama-sama memiliki akses pada informasi yang lengkap tentang keadaan permainan, sehingga tidak ada informasi yang tertutup bagi lawan mainnya. (2005, gunawan)

Para pemain diberi label Max dan Min. Max bergerak dari *node* berbentuk persegi dan Min pergerakannya dari *node* berbentuk *node* lingkaran. Diberi nilai dalam posisi setiap pergerakan sesuai dengan properti minimax.



Gambar 2.2 Game Tree pada *Tic Tac Toe*

(Sumber: Wim Pijls and Arie de Bruin)

Lebih jelas dapat ditunjukkan dari tabel 2.1 berikut :

Tabel 2.1. Jumlah Kemungkinan Langkah

Langkah	Kemungkinan	Total
1	9	9
2	9x8	72
3	9x8x7	504
4	9x8x7x6	3024
5	9x8x7x6x5	15120
6	9x8x7x6x5x4	60480
7	9x8x7x6x5x4x3	181440
8	9x8x7x6x5x4x3x2	362880

2.4 C++

2.4.1 Sejarah C++

Tahun 1978, Brian W. Kerninghan & Dennis M. Ritchie dari AT & T Laboratories mengembangkan bahasa B menjadi bahasa C. Bahasa B yang diciptakan oleh Ken Thompson sebenarnya merupakan pengembangan dari bahasa BCPL (*Basic Combined Programming Language*) yang diciptakan oleh Martin Richard. Sejak tahun 1980, bahasa C banyak digunakan pemrogram di Eropa yang sebelumnya menggunakan bahasa B dan BCPL. Dalam perkembangannya, bahasa C menjadi bahasa paling populer diantara bahasa lainnya, seperti *PASCAL*, *BASIC*, *FORTRAN*. Tahun 1989, dunia pemrograman C

mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh *American National Standards Institute* (ANSI). Bahasa C yang diciptakan Kerninghan & Ritchie kemudian dikenal dengan nama ANSI C. Mulai awal tahun 1980, *Bjarne Stroustrup* dari *AT & T Bell Laboratories* mulai mengembangkan bahasa C. Pada tahun 1983, lahirlah secara resmi bahasa baru hasil pengembangan C yang dikenal dengan nama C++. Bahasa ini bersifat kompatibel dengan bahasa pendahulunya yaitu C.Pada mulanya C++ disebut dengan "*a better C*". Nama C++ sendiri diberikan oleh Rick Mascitti pada musin panas 1983.

(Sumber : http://prita.staff.gunadarma.ac.id**)**

2.4.2 Perkembangan C++

Pada tahun 1990, Referensi *The Annotated* C++ dirilis. Pada tahun yang sama, Turbo c++ menambahkan kebanyakan *library* tambahan yang akan memiliki dampak yang besar terhadap yang besar terhadap pengembangan C++'s. Meskipun terakhir rilis stabil Turbo C++'s pada tahun 2006, *compiler* masih banyak digunakan.

Pada tahun 1998, panitia C++ menetapkan standar internasional yang pertama untuk C++ ISO/IEC 14882: 1998 yang dikenal sebagai C++ 98. Referensi beranotasi C++ memiliki pengaruh besar dalam pengembangan standar C++. Pada tahun 2003 panitia C++ menanggapi beberapa permasalahan yang dilaporkan dari standar 1998 dan merevisi sesusai permsalahan yang ada. Hasil dari revisi tersebut dijuluki C++ 03.

Pada tahun 2005, panitia C++ merilis laporan teknis (dijuluki TR1) secara rinci dari berbagai fitur untuk menambah standar terbaru C++. Standar baru

itu dijuluki C++ 0x yang rencananya dapat dirilissebelum akhir dekade pertama. Nmaun, standar baru tersebut tidak akan dirilis sampai pertengahan tahun 2011.

Pada tahun 2011, C++ standar baru telah dirilis (dijuluki C++ 11). Proyek meningkatkan *library* membuat dampak yang cukup besar pada standar baru. Beberapa fitur baru terdapat penunjang ekspersi reguler, sebuah *library* pengancakan yang komperhensif, Sebuah *library* waktu C++ terbaru, sebuah penunjang *atomics*, sebuah *library* rangkaian standar, sebuah *syntax* perulangan FOR baru yang menyediakan fungsi mirip dengan perulangan *foreach* di dalam bahasa lain. (sumber : http://www.cplusplus.com/info/history)

2.5 Pointer

Variabel merupakan suatu nilai yang disimpan dalam *memory* yang dapat diakses dengan *identifier*. Variabel ini sesunggunhnya disimpan pada suatu alamat didalam *memory*. Dimana setiap alamat *memory* akan berbeda dengan yang lainnya (unik).

Arti pointer dalam bahasa sehari-hari adalah putunjuk atau bisa di bilang penentu atau pointer secara sederhana bisa diartikan sebagai tipe data yang nilainya mengarah pada nilai yang terdapat pada sebuah area memori (alamat memori).namun dalam Dalam bahasa C, Pointer bisa berfungsi sebagai variabel array berarti pointer sebagai penunjuk elemen array ke-0 dalam variabel C.

2.5.1 Operator Alamat (Address operator (&))

Pada saat pendeklarasian variable, *user* tidak diharuskan menentukan lokasi sesungguhnya pada memory, hal ini akan dilakukan secara otomatis oleh

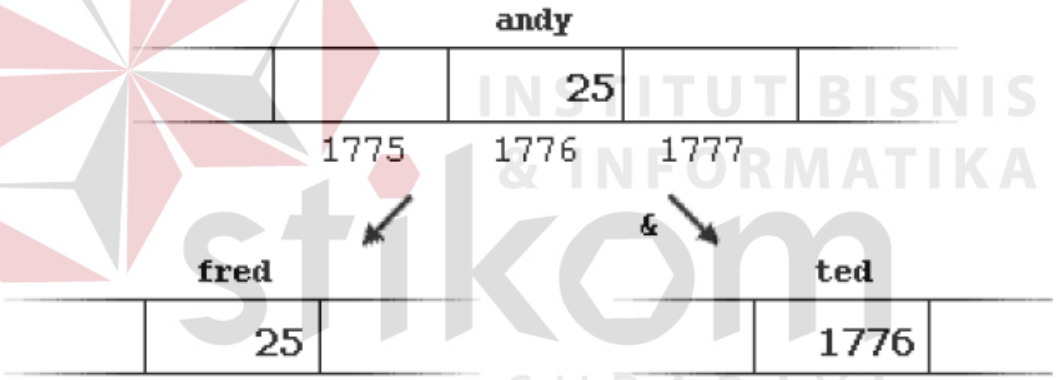
kompiler dan *operating sysem* pada saat *run-time*. Jika ingin mengetahui dimana suatu variable akan disimpan, dapat dilakukan dengan memberikan tanda *ampersand* (&) didepan variable , yang berarti ***address of***.

Contoh : ted = &andy

Akan memberikan variable **ted** alamat dari variable **andy**, karena variable **andy** diberi awalan karakter *ampersand* (&), maka yang menjadi pokok disini adalah alamat dalam *memory*, bukan isi variable. Misalkan **andy** diletakkan pada alamat **1776** kemudian dituliskan instruksi sbb :

andy = 25, fred = andy dan ted = &andy.

Berikut hasil dari intruksi tersebut :



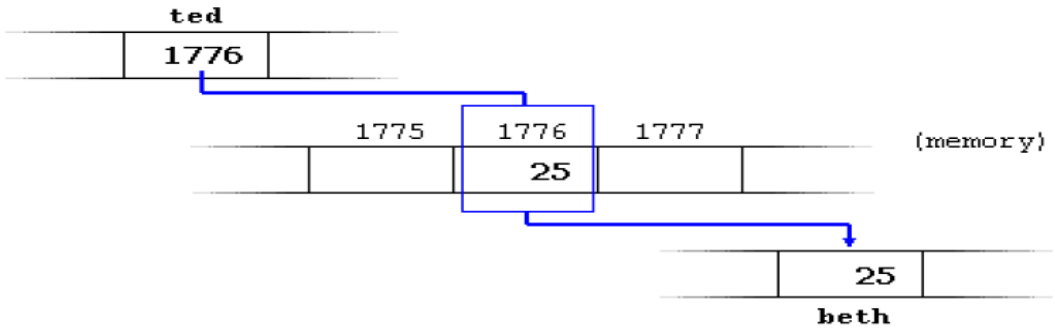
Gambar 2.3 Hasil Intruksi Pointer Operator Alamat

2.5.2 Operator Reference(*)

Dengan menggunakan pointer, kita dapat mengakses nilai yang tersimpan secara langsung dengan memberikan awalan operator *asterisk* (*) pada *identifier* pointer, yang berarti ***"value pointed by"***. Contoh :

beth = *ted

(dapat dikatakan:"beth sama dengan nilai yang ditunjuk oleh ted") **beth** = **25**, karena **ted** dialamat **1776**, dan nilai yang berada pada alamat **1776** adalah **25**.



Gambar 2.4 Hasil Intruksi Pointer Operator Operance

(Sumber : http://prita.staff.gunadarma.ac.id**)**

2.5.3 Macam–macam Pointer

a. Pointer Bertiper Void

Pada C++ terdapat pointer yang dapat menunjuk ke tipe data apapun, pointer semacam ini dideklarasikan dengan tipe void sehingga sering dikenal dengan istilah Void Pointer. Berikut ini contoh *listing* program yang menggunakan void pointer.

b. Pointer Bertiper Void

Elemen-elemen array biasanya diakses melalui indeksnya, sebenarnya ada cara lain yang lebih efisien, yaitu dengan menggunakan pointer. Pointer semacam ini disebut dengan istilah pointer aritmetika. Konsep dasar dari pointer aritmetika ini adalah melakukan operasi aritmetika terhadap variabel yang bertipe pointer.

c. Pointer NULL

Pada saat program dijalankan, pointer akan menunjuk ke alamat acak pada memori, sehingga diperlukan inisialisasi agar hal tersebut tidak terjadi.

Dalam C++ terdapat sebuah cara untuk membuat pointer tidak menunjuk ke alamat manapun, yaitu dengan mengisikan ponter tersebut dengan nilai NULL. Karena hal inilah maka pointer tersebut sering dinamakan pointer NULL (NULL Pointer).Sebagai contoh kita mempunyai pointer p, dan kita ingin melakukan inisialisasi pada pointer tersebut dengan nilai NULL.

(Sumber : http://prita.staff.gunadarma.ac.id**)**

2.6 Linked List

Link List adalah sejumlah objek yang di *link* atau dihubungkan satu dengan yang lainnya sehingga membentuk suatu list. Sedangkan objek itu sendiri adalah **gabungan beberapa elemen** data(variabel) yang dijadikan satu kelompok atau ***structure*** atau ***record*** yang dibentuk dengan perintah ***struct.*** Tiap-tiap elemen dapat memiliki tipe data tersendiri yang berbeda denga tipe data elemen lain. Untuk menyambungkan objek satu dengan objek lainnya, diperlukan paling tidak sebuah variabel yang 'bertipe' pointer. Variabel pointer tersebut merupakan salah satu variabel dalam struktur objek. (sumber : Moh,Sjukani,2012)

Secara teori *linked list* adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer. Dikatakan *single (singly) linked* apabila hanya ada satu pointer yang menghubungkan setiap *node. Single* artinya *field* pointer nya hanya satu buah saja dan satu arah. Senarai berkait adalah struktur data yang paling dasar. Senarai berkait terdiri atas sejumlah unsur-unsur dikelompokkan, atau terhubung, bersama-sama di suatu deret yang spesifik. Senarai berkait bermanfaat di dalam memelihara koleksi-koleksi data, yang serupa dengan array/larik yang sering digunakan. Bagaimanapun juga, senarai berkait memberikan keuntungan-

keuntungan penting yang melebihi array/larik dalam banyak hal. Secara rinci,senarai berkait lebih efisien di dalam melaksanakan penyisipan-penyisipan dan penghapusan-penghapusan. Senarai berkait juga menggunakan alokasi penyimpanan secara dinamis, yang merupakan penyimpanan yang dialokasikan pada *runtime*. Karena di dalam banyak aplikasi,ukuran dari data itu tidak diketahui pada saat kompile, hal ini bisa merupakan suatu atribut yang baik juga. Setiap *node* akan berbentuk struct dan memiliki satu buah *field* bertipe *struct* yang sama, yang berfungsi sebagai pointer Dalam menghubungkan setiap node, kita dapat menggunakan cara *first-create-first-access* ataupun *first-create-last-access*. Yang berbeda dengan deklarasi struct sebelumnya adalah satu *field* bernama *next* , yang bertipe *struct* tnode. Hal ini sekilas dapat membingungkan. Namun, satu hal yang jelas, variabel *next* ini akan menghubungkan kita dengan node di sebelah kita, yang juga bertipe struct tnode. Hal inilah yang menyebabkan *next* harus bertipe *struct* tnode.

Bentuk Umum *Link List* :

```
Typdef struct telmtlist
{
        Infotype info;
        Addres next;
} elmtist;
```

Infotype : Sebuah tipe terdefinisi yang menyimpan nformasi sebuah elemen list.

Next : Alamat dari emelen berikutnya(suksesor).

(sumber : https://id.scribd.com/doc/28723767/Modul-Linked-List-c (Teknik Elektro– Universitas Negeri Malang – 2010))

2.6.1 Single Linked List

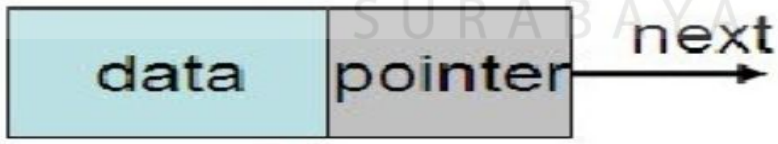
Senarai berkait yang paling sederhana, di mana unsur-unsur terhubung oleh suatu pointer.Struktur ini mengizinkan senarai dilintasi dari elemen pertama sampai elemen terakhir.

Abstraksi Tipe Data Single Linked List Non Circular

Pembuatan struct bernama tnode berisi 2 *field*, yaitu field data bertipe integer dan *field next* yang bertipe pointer dari tnode.

Deklarasi Node SLL :

```
Struct tnode
{
    Int data;
    Struct tnode *next;
}
```

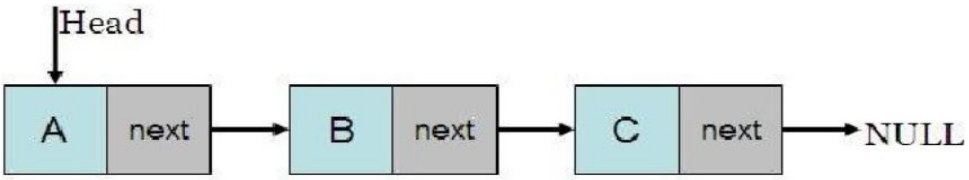


2.5 Sebuah Node pada SLL

Seperti yang diungkapkan sebelumnya, bahwa akan dibuat *Single Linked List* (SLL) dengan cara *first-create-first-access*. Dengan cara ini, node yang dibuat pertama akan menjadi *head*. *Node-node* yang dibuat setelahnya akan menjadi *node-node* pengikut. Untuk mempermudah pengingatan, ingatlah gambar anak panah yang mengarah ke

kanan. Headakan berada pada pangkal anak panah, dan *node-node* berikutnya akan berbaris ke arah bagian anak panah yang tajam.

(sumber: https://id.scribd.com/doc/28723767/Modul-Linked-List-c (Teknik Elektro– Universitas Negeri Malang – 2010))

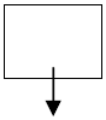


2.6 Single Link List

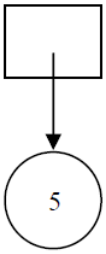
(sumber : https://id.scribd.com/doc/28723767/Modul-Linked-List-c (Teknik Elektro– Universitas Negeri Malang – 2010)

2.7 Binary Tree

Binary Tree adalah struktur data yang hampir mirip juga dengan *Linked List* untuk menyimpan koleksi dari data. *Linked List* dapat dianalogikan sebagai rantai linier sedangkan *Binary Tree* bisa digambarkan sebagai rantai tidak linier. *Binary Tree* dikelompokkan menjadi *unordered Binary Tree* (tree yang tidak berurut) dan *ordered Binary Tree* (tree yang terurut). *Binary Tree* dapat digambarkan berdasarkan kondisinya, sebagai berikut:



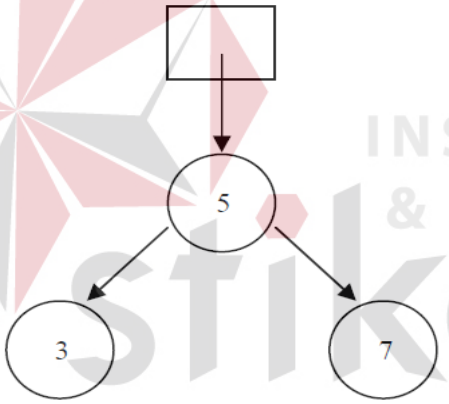
Binary Tree kosong



Pointer ke akar (*root*) dari tree

Binary Tree sebagai *root* sekaligus sebagai daun (*leaf*)

2.7 Binary Tree



Pointer ke root dari tree

Root dari tree

Leaf dari tree

2.8 Gambaran dari *Binary Tree* yang Terdiri dari 3 (tiga) *Node*

(Sumber : http://www.informatika.unsyiah.ac.id/tfa/ds/bst.pdf)