BAB II

LANDASAN TEORI

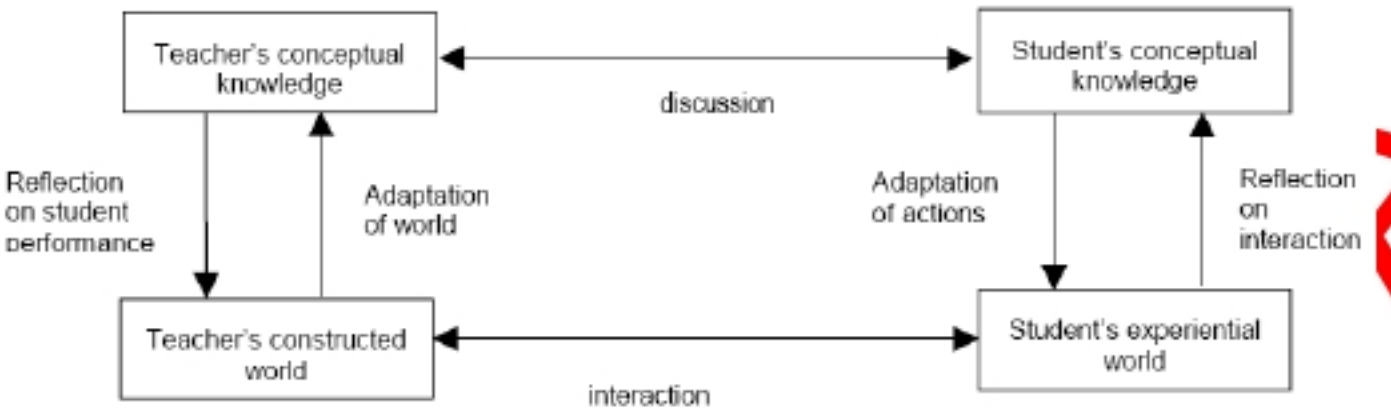
2.1 Pembelajaran Jarak Jauh

Pembelajaran jarak jauh adalah sekumpulan metode pengajaran dimana aktivitas pengajaran dilaksanakan secara terpisah dari aktivitas belajar. Pemisah kedua kegiatan tersebut dapat berupa jarak fisik, misalnya karena peserta ajar bertempat tinggal jauh dari lokasi institusi pembelajaran. Pemisah dapat pula jarak non-fisik yaitu berupa keadaan yang memaksa seseorang yang tempat tinggalnya dekat dari lokasi institusi pembelajaran namun tidak dapat mengikuti kegiatan pembelajaran di institusi tersebut.

Pengertian jarak jauh adalah tidak terjadinya kontak dalam bentuk tatap muka langsung antara guru dan siswa ketika proses belajar mengajar terjadi. Dengan demikian, pembelajaran jarak jauh adalah komunikasi dua arah yang dijembatani oleh media seperti surat, telepon dan *internet*. Mekanisme sistem pembelajaran jarak jauh pada umumnya memaksa lembaga penyelenggara pembelajaran jarak jauh itu sendiri mempercayai akan kejujuran dan kemandirian siswa (Anggoro, 2002). Sistem pembelajaran jarak jauh ini dapat mengatasi beberapa masalah yang ditimbulkan akibat keterbatasan tenaga pengajar yang berkualitas.

2.2 Proses Pembelajaran

Secara umum proses pembelajaran yang terjadi antara guru dengan murid dapat digambarkan sebagai berikut :



Gambar 2.1 Proses Pembelajaran

a) Discussion

Diskusi yang terjadi antara guru dengan murid selama proses pembelajaran. Diskusi sangat penting perannya bagi murid untuk proses akuisisi pengetahuan yang diperoleh dari guru.

b) Interaction

Interaksi antara guru dengan murid dan sebaliknya. Interaksi yang dapat terjadi adalah hal – hal seperti *rehearsal, self-paced learning and practice*.

c) Adaptation

Feedback dari guru atau murid setelah memperoleh respon dari salah satu pihak.

d) Reflection

Refleksi dapat terjadi ketika guru mendengarkan suatu permainan dari murid *(reflection on student performance)*. Secara umum refleksi merupakan suatu pemikiran sistematis yang dilakukan oleh guru tentang apa yang diperbuat oleh murid.

2.3 NMP (Network Musical Performance)

NMP dapat terjadi ketika suatu kelompok musisi berada pada lokasi yang berbeda secara fisik, kemudian berinteraksi melalui jaringan komputer untuk memainkan suatu komposisi seakan-akan mereka terletak pada satu ruangan yang sama (Lazzaro dan Wawrzynek, 2001). NMP juga memiliki keuntungan apabila diterapkan sebagai *remote live performance, music collaboration* dan *musical instruction* karena tidak adanya batasan lokasi secara fisik. *Musical instruction* itu sendiri merupakan suatu pembelajaran akan suatu jenis alat musik. Untuk memungkinkan terjadinya NMP, ada 2 (dua) cara yang dapat digunakan untuk melakukan proses transmisi data di dalam jaringan:

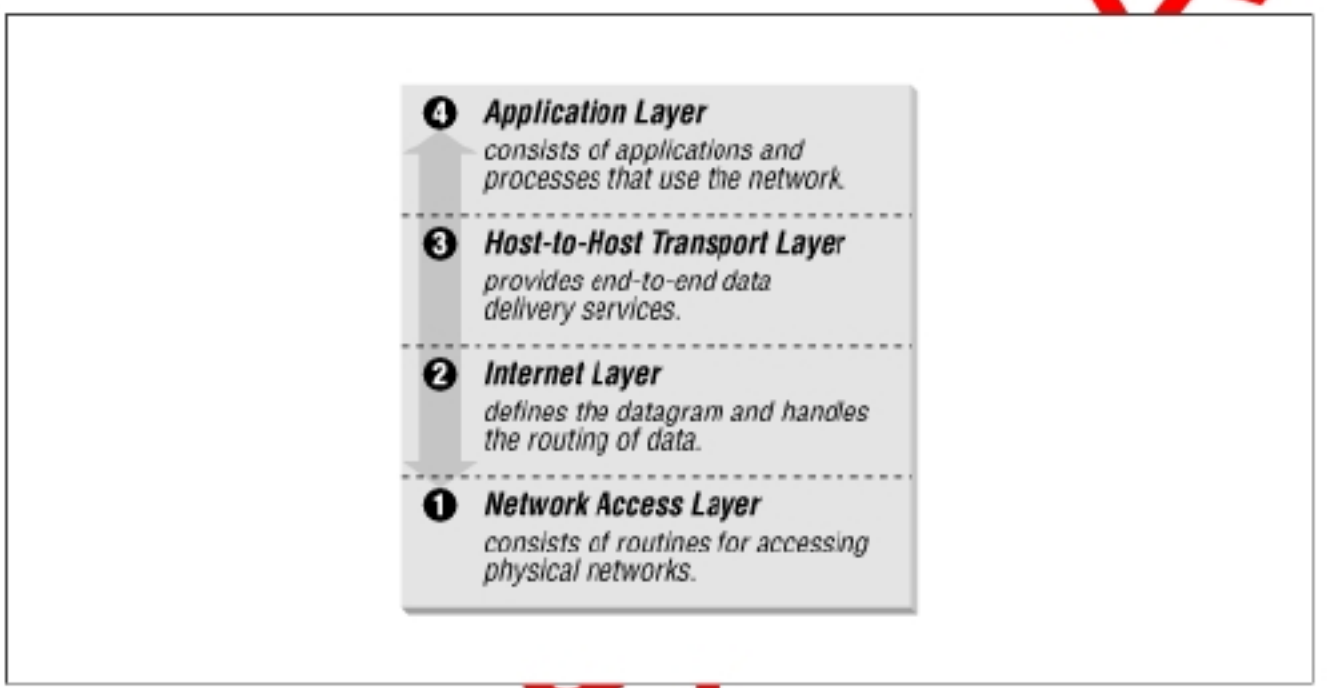
1. Menggunakan *bandwidth* yang sangat besar untuk mengirim sampel audio yang memiliki format digital seperti *wav, mp3* dan *ogg*.
2. Menggunakan *bandwidth* yang relatif lebih kecil untuk mengirim *musical representative data*.

Cara kedua memiliki keuntungan karena menggunakan *bandwidth* yang lebih kecil sehingga *latency* dapat diminimalkan (Young dan Fujinaga, 1999).

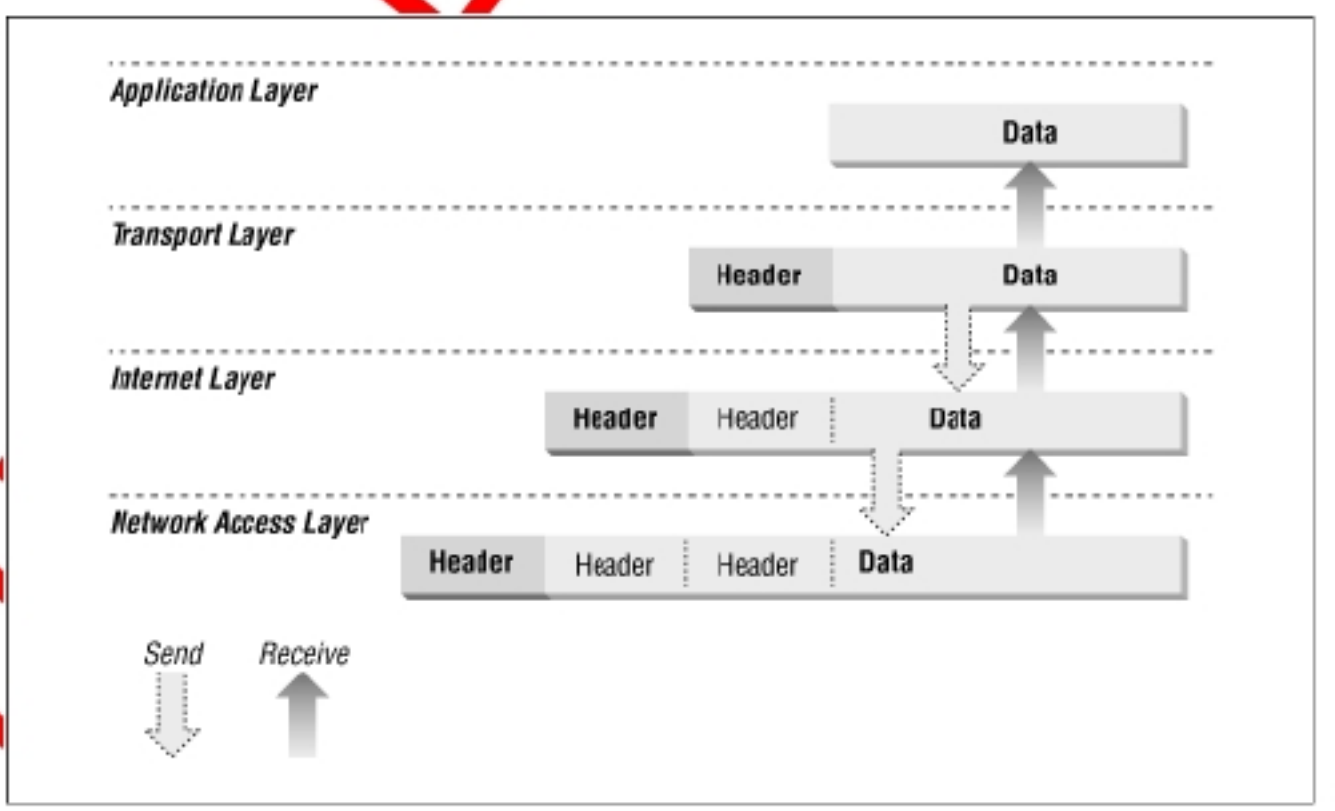
2.4 TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP adalah sekumpulan protokol yang digunakan untuk berkomunikasi di internet, TCP/IP merupakan gabungan singkatan dari *Transmission Control Protocol* dan *Internet Protocol* yang menjadi fundamental dari bahasa komunikasi internet. Konsep komunikasi dari protokol ini dilukiskan sebagai tumpukan (*stack*) berlapis – lapis yang lebih dikenal dengan OSI (*Open System Interconnect*) yang menjadi model dari bagaimana seharusnya suatu komunikasi dilaksanakan, berinteraksi antara perangkat lunak dengan perangkat

keras jaringan. Pendekatan mengenai lapisan OSI model ini didalam implementasi nyata dalam berkomunikasi digantikan oleh model baru yang lebih praktis dan berguna, pengganti tersebut lebih dikenal dengan model *Internet TCP/IP Protocol Suite*, yang mempunyai model lapisan mirip dengan OSI model namun hanya berjumlah 4 lapisan (Hunt, 1997).



Gambar 2.2 TCP/IP Protocol Architecture



Gambar 2.3 Data Encapsulation

Jika suatu protokol menerima data dari protokol lain di *layer* atasnya, ia akan menambahkan informasi tambahan miliknya ke data tersebut, informasi ini memiliki fungsi yang sesuai dengan fungsi protokol tersebut. Setelah itu, data ini diteruskan lagi ke protokol pada layer di bawahnya. Hal yang sebaliknya terjadi jika suatu protokol menerima data dari protokol lain yang berada pada *layer* di bawahnya. Jika data ini dianggap valid, protokol akan melepas informasi tambahan tersebut untuk kemudian meneruskan data itu ke protokol lain yang berada pada *layer* di atasnya.

2.4.1 Application Layer

Lapisan ini merupakan lapisan tertinggi dalam protokol TCP/IP, lapisan ini merupakan perantara antara layanan aplikasi dengan proses didalamnya, yang didefinisikan untuk saling berkomunikasi membentuk hubungan antar layanan dengan layanan yang lainnya. Contoh dari layanan ini diantaranya adalah SSH *(Secure Shell)*, HTTP *(Hyper Text Transfer Protocol)*, DNS *(Domain Name Service)* dan SMTP *(Simple Mail Transfer Protocol)*.

2.4.2 Transport Layer

Transport layer menyediakan data transfer yang efektif dan efisien melalui mekanisme *retransmission* dan pelacakan paket data dari suatu link. Contoh aktual dari lapisan ini adalah protokol TCP, yang mempunyai ciri *connection-oriented, reliable protocol, byte stream service. Connection oriented* berarti sebelum melakukan pertukaran data, dua aplikasi pengguna *TCP* harus melakukan hubungan *(handshake)* terlebih dahulu. *Reliable* berarti *TCP*

menerapkan proses deteksi kesalahan paket dan retransmisi. *Byte stream service* berarti paket dikirimkan dan sampai ke tujuan secara berurutan.

2.4.3 Internet Layer

Lapisan ini berada diatas lapisan akses network dalam struktur hirarki protokol. Protokol Internet disebutkan sebagai jantung TCP/IP dan memiliki peran penting dalam lapisan Internet. IP membuat paket dasar untuk servis pengiriman data *(delivery)*. Semua protokol pada lapisan diatas dan dibawah IP, memakai protokol Internet untuk pengiriman datanya. Lapisan ini menggunakan algoritma *routing* untuk menentukan kemana datagaram harus dikirim.

2.4.4 Network Access Layer

Lapisan ini merupakan lapisan terendah dalam protokol TCP/IP. Protokol dalam lapisan ini memungkin sistem untuk melakukan pengiriman dan penerimaan data ke atau dari perangkat lainnya yang tersambung ke jaringan komputer. Dengan adanya protokol dalam lapisan ini maka sebuah datagram IP bisa dikirimkan lewat jaringan komputer. Protokol di lapisan ini harus mengenali jaringan komputernya (seperti struktur paket datanya, alamatnya) agar format data bisa terjaga dan disesuaikan dengan kondisi jaringan komputer. Pada lapisan ini terjadi enkapsulasi *datagram* IP menjadi sebuah *frame* yang akan dikirimkan ke jaringan komputer dan memetakan alamat IP ke alamat fisik yang dipakai dalam jaringan komputer.

2.5 MIDI (Musical Instrument Digital Interface)

MIDI (*Musical Instrument Digital Interface*) adalah standar industri / perusahaan pembuat peralatan musik untuk berkomunikasi / berhubungan antara

pembuat alat yang berbeda. Standar MIDI diangkat oleh industri peralatan musik pada tahun 1983 (White, 2001). MIDI dapat digunakan untuk menyelaraskan banyak macam peralatan musik yang berbeda seperti *synthesizer, digital samplers, drum machines, keyboard controller, guitar controller, wind instrument controller* dan lain sebagainya. MIDI umumnya memiliki 16 *channel* dimana masing-masing *channel* dapat mengirimkan informasi data digital seperti:

a. Kapan nada itu bunyi atau tidak.
b. Sensitivitas tekanan tuts dari nada.
c. Perubahan warna suara (*Sound Patch*).
d. Peraba kekerasan / *Velocity*.
e. Besarnya Suara / Volume, *Pitch Bends*.

Suatu file MIDI dapat memiliki lebih dari satu event. Satu event terdiri dari waktu (*time*), tipe pesan (*message type*) dan suatu nomor parameter yang spesifik (*data bytes*).

Beberapa contoh event, yaitu :

Tabel 2.1 MIDI Event

Tipe	Nama	Batasan
80	Note Off	3 Bytes (Channel, Note, Velocity)
90	Note On	3 Bytes (Channel, Note, Velocity)
C0	Program Change	2 Byte (Channel, Program Number)

Secara garis besar, ada 12 nada yang dapat digunakan. Dan antara setiap nada tersebut mempunyai jarak masing masing yang disebut dengan interval. Berikut ini merupakan susunan tabel nada yang dapat digunakan :

Tabel 2.2 MIDI Note

Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

Berikut ini merupakan deklarasi struktur yang dibutuhkan untuk mengirimkan *note on event* pada MIDI:

```
struct RawMIDI
{
  unsigned char channel;
  unsigned char note;
  unsigned char velocity;
}raw;
raw.channel = 0x90;
raw.note = 60;
raw.velocity = 90;
write(fd_out, &raw, 3);
```

2.6 Latency

Latency didefinisikan sebagai *delay* waktu ketika *stream* data dikirim dan ketika *stream* itu dihadirkan pada *end user*. Berdasarkan pada hasil penelitian yang menyebutkan bahwa batas *real time* area audio, untuk aplikasi biasa memiliki sensitivitas perbedaan waktu sekitar 7 *millisecond*, dan 5 *millisecond* untuk aplikasi profesional (Phillips, 2000), maka sangat diperlukan untuk mengetahui bahwa *latency* yang dihasilkan dari sistem tidak akan melewati batas dari *real time* pada area audio. Fungsi *gettimeofday()* pada sistem operasi Linux dapat digunakan untuk mengetahui berapa lama suatu proses telah berhasil menyelesaikan pekerjaannya. Fungsi ini dapat mengukur durasi proses hingga dalam ukuran *millisecond*. Fungsi inilah yang digunakan untuk mengukur *read* dan *write latency* pada MIDI *device*, *gettimeofday()* menggunakan *struct timeval* untuk menyimpan hasil nilai balik. *Struct timeval* didefinisikan sebagai berikut:

```
struct timeval {
        time_t      tv_sec;     /* seconds */
        suseconds_t  tv_usec;  /* microseconds */
};
struct timeval before, after;
gettimeofday(&before, NULL);
write(fd_out, &raw[0], 3);
gettimeofday(&after, NULL);
```

2.6.1 Write Latency

Waktu yang dibutuhkan untuk mengirimkan data. Dari data – data tersebut diketahui rata-ratanya yaitu sebesar 0.03 *millisecond*.

Tabel 2.3 Write latency

No	Paket (byte)	Waktu (millisecond)
1	30	0.04
2	30	0.01
3	30	0.03
4	30	0.01
5	30	0.03
6	30	0.03
7	30	0.03
8	30	0.03
9	30	0.03
10	30	0.03

2.6.2 Read Latency

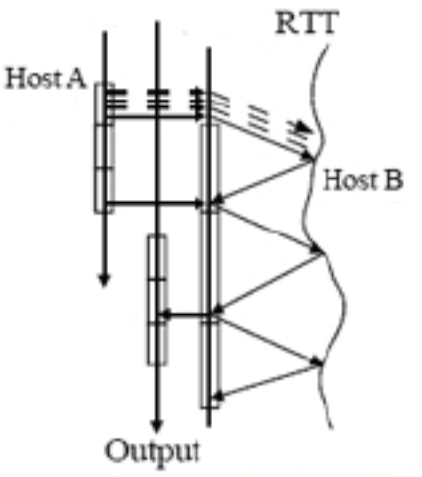
Waktu yang dibutuhkan untuk menerima data. Dari data – data tersebut diketahui rata-ratanya yaitu sebesar 0.02 *millisecond*.

Tabel 2.4 Read latency

No	Paket (byte)	Waktu (millisecond)
1	30	0.03
2	30	0.03
3	30	0.03
4	30	0.01
5	30	0.01
6	30	0.01
7	30	0.01
8	30	0.03
9	30	0.01
10	30	0.01

2.6.3 Round Trip Time (RTT)

RTT adalah waktu yang dibutuhkan agar suatu paket dapat diterima dari satu host ke host yang lain.



Gambar 2.4 Round Trip Time

Berikut ini merupakan waktu yang dibutuhkan untuk menerima paket dari host A (192.168.0.1) ke host B (192.168.0.2) yang terhubung dalam jaringan melalui *switch* dengan kabel RJ-45.

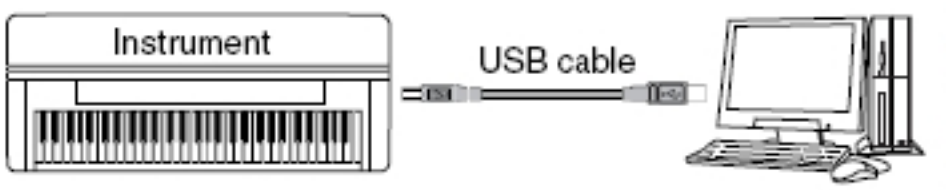
Tabel 2.5 Round trip time

No	Paket (byte)	Waktu (millisecond)
1	208	0.318
2	208	0.314
3	208	0.313
4	208	0.307
5	208	0.317
6	208	0.317
7	208	0.298
8	208	0.314
9	208	0.317
10	208	0.308

Dari data – data tersebut diketahui rata-ratanya yaitu sebesar 0.31 *millisecond*.

2.7 Linked List

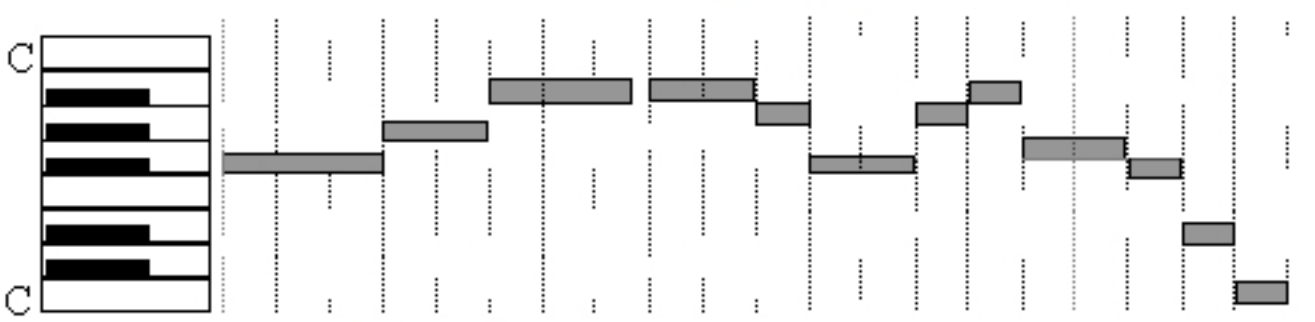
Proses penanganan input dilakukan dengan menggunakan keyboard MIDI controller yang terhubung ke komputer melalui *Universal Serial Bus (USB).* Seperti yang diilustrasikan pada gambar berikut.



Gambar 2.5 MIDI controller via USB

Apabila sebagai contoh input yang terjadi memiliki urutan *(sequence)* :

F4 – G4 – B4 – B4 – A4 – F4 – A4 – B4 – G4 – F4 – E4 – C4



Gambar 2.6 Representasi Input

seperti yang tampak pada gambar diatas, maka aplikasi akan menyimpan data – data tersebut kedalam struktur data *linked list*, struktur data ini dipilih karena *linked list* dapat mengimplementasikan *growable array of objects*. Seperti pada *array*, *linked list* dapat terdiri dari komponen yang dapat diakses melalui *index integer*. Selain itu ukuran dari *linked list* dapat bertambah atau berkurang sesuai kebutuhan setelah suatu *linked list* telah dibuat (Deitel, 2004).