

KONTRAK PRAKTIKUM

Nama Mata Kuliah	: Praktikum Struktur Data
Kode Mata Praktikum	: TKC 119
Mata Kuliah Prasyarat	: Struktur Data
Hari Pertemuan / Jam	:
Tempat Pertemuan	: Lab. Pemrograman

Gambaran Umum :

Praktikum ini merupakan praktek dari mata kuliah Struktur Data dan Algoritma Pemrograman yang meliputi array, stack, searching, sorting, pointer, dan link list. Pada praktikum ini mahasiswa melakukan pemrograman penggunaan struktur data array, stack, linked list, yang banyak digunakan untuk pemecahan permasalahan dalam pemrograman. Selain itu juga dilakukan pemrograman beberapa algoritma sorting dan searching. Praktikum disertai dengan latihan-latihan menyelesaikan permasalahan dengan struktur data yang tepat.

Mahasiswa diharapkan dapat:

- Menyelesaikan masalah menjadi sebuah algoritma (langkah-langkah) yang akan dijalankan oleh komputer, kemudian mengimplementasikannya menjadi sebuah program komputer
- Merepresentasikan data yang digunakan dalam pemrograman (baik data input atau data output) dengan struktur data yang tepat seperti array, stack, linked list.
- Mengetahui & membandingkan macam-macam algoritma dalam proses pengurutan dan pencarian dan dapat menentukan algoritma yang digunakan dalam permasalahan pemrograman yang diselesaikannya.

Alur bagi peserta praktikum :

1. Peserta praktikum menerima dan kemudian mempelajari modul praktikum.
2. Peserta praktikum mengerjakan tugas pendahuluan yang diberikan.
3. Peserta praktikum melakukan asistensi tugas pendahuluan tersebut. Asistensi ini digunakan sebagai bagi asisten untuk menilai kesiapan peserta juga berfungsi sebagai ajang diskusi peserta praktikum atas kesulitan yang dialaminya.
4. Peserta praktikum mendemokan implementasi pada asisten. Penilaian yang dilakukan oleh asisten bersifat WISIWYG (What I see Is What You Get)

Peraturan Praktikum :

1. Praktikan harus menghadiri setiap sesi praktikum, tidak hadir tanpa keterangan akan menyebabkan hak menjadi praktikan gugur (nilai E).
2. Tugas pendahuluan diserahkan sebelum praktikum dimulai.
3. Laporan praktikum dianggap sah apabila praktikan telah melakukan asistensi sesuai dengan materi praktikum.
4. Laporan resmi adalah gabungan dari semua laporan praktikum tiap sesi.

Grade Penilaian Praktikum :

1. Tugas pendahuluan 15%
2. Kehadiran dan tugas praktikum 30%
3. Asistensi dan laporan praktikum 25%
4. Pretest dan Post Test (15%)
5. Laporan resmi 15%

MODUL I

TIPE DATA ARRAY

Tujuan Praktikum :

1. Mahasiswa dapat memahami konsep tipe data array.
2. Mahasiswa dapat mengimplementasikan tipe data array pada suatu permasalahan.

Tugas Pendahuluan :

1. Jelaskan konsep struktur dan deklarasi struktur untuk array berdimensi satu dan dua !
2. Buatlah algoritma untuk matrik (mxn) 15x15. Inputkan data-datanya dan tampilkan :
 - a. Semua elemen matrik tersebut !
 - b. Tampilkan elemen-elemen matrik yang $m = n$!
3. Buat algoritma untuk perkalian matrik.

Landasan Teori:

Selama ini kita sudah mengenal variabel bertipe **real, integer, char, string**. Satu variabel bisa menampung satu nilai dengan tipe yang telah kita definisikan sebelumnya. Misalnya kita mendefinisikan suatu variabel dengan nama *i* yang bertipe integer, maka variabel *i* ini dapat menampung satu buah data dengan tipe integer.

Misalkan kita masukkan nilai 5, maka variabel *i* akan berisi nilai 5, apabila kemudian variabel ini diisi lagi dengan nilai 8, maka isi variabel yang tadinya bernilai 5 ditumpuk dengan nilai 8 sehingga isi dari variabel *i* sekarang menjadi 8.

Yang jadi permasalahan adalah kita memiliki sejumlah data yang mempunyai tipe data yang sama, dan kesemuanya ingin disimpan didalam variabel, maka salah satu cara pemecahannya adalah dengan menggunakan satu buah variabel untuk satu jenis data. Cara ini memang sepertinya menyelesaikan masalah tetapi coba bayangkan apabila kita mempunyai data sejumlah seratus buah, maka kita harus mendeklarasikan variabel sejumlah seratus buah nama variabel dengan tipe yang sama, ini merupakan suatu pemborosan sumber daya yang ada, dan akan memakan banyak sumber daya memori dari komputer.

Tetapi untungnya didalam Pascal ada satu jenis tipe data yang disebut sebagai array. Dengan array ini kita dapat menampung sejumlah data dengan tipe yang sama didalam satu buah variabel.

Konsep Dasar

Didalam array, ada beberapa definisi yang perlu untuk dikenal, sehingga konsep array benar-benar bisa dipahami, istilah-istilah tersebut antara lain :

1. Array adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama variabel yang sama.
2. Setiap data yang ada didalam array disebut sebagai elemen array.
3. Setiap elemen dari array dapat diakses dengan pengenalnya adalah indeksinya. Setiap elemen mempunyai indeks tersendiri. Satu indeks mewakili satu elemen.
4. Karena ada indeks, array juga dikenal dengan variabel berindeks.

Deklarasi Array

Bentuk deklarasi dari array adalah :

Var nama : array [indeks] of tipe

Dimana **var**, **array** dan **of** adalah reserved word milik pascal. Nama adalah nama pengenalan dari variabel bertipe array. Indeks adalah batasan indeks dari variabel ini yang menggambarkan jumlah elemen dari variabel array tersebut.

Contoh-contoh dari deklarasi array :

Var nilai : array [1..10] of integer

Definisi array diatas berarti kita mendeklarasikan sebuah variabel dengan **tipe array** yang mempunyai nama variabel nilai, yang memiliki elemen sejumlah 10 buah dengan tipe datanya adalah integer. Dengan indeks yang mempunyai batas bawah 1 dan batas atas adalah 10.

Statemen pemberian nilai pada array

Contoh pemberian nilai pada var yang melibatkan variabel array

```
Var nilai      : array [1..100] of integer;  
    hasil      : array [1..10] of real;  
    sementara : array [1..20] of real;  
    jumlah     : integer;  
    temp       : real;
```

.
. .
. .
. .

Jumlah := nilai [4];

Statement diatas berarti variabel jumlah diisi dengan isi dari variabel nilai pada elemen keempat.

Array Multidimensi

Array juga dapat didefinisikan untuk dimensi lebih dari satu atau multidimensi, bisa dimensi dua, dimensi tiga, dan sebagainya. Array dua dimensi dapat mewakili

penggambaran sebuah tabel ataupun matrik. Bentuk deklarasi dari sebuah matrik multi dimensi adalah sebagai berikut :

NamaArray : array [indeks1] of tipearray [indeks2] of tipearray

Atau dapat juga ditulis dengan

NamaArray : array [indeks1, indeks2] of tipearray

Contoh deklarasi dari array multidimensi:

Var

Tabel : array [1..3, 1...2] of integer;

Tugas Praktikum :

(Asisten yang menentukan hanya 2 soal saja yang dikerjakan oleh praktikan, bisa untuk masing-masing praktikan soalnya berbeda)

1. Buatlah program untuk menghitung nilai rata-rata yang dimasukkan user dengan banyak data sesuai dengan yang dimasukkannya.
2. Buatlah program untuk sebuah fungsi bernama test polindrom untuk menguji apakah sebuah string bersifat polindrom, artinya dibaca dari kiri sama dengan dibaca dari kanan. Fungsi akan menghasilkan nilai 1 jika benar, dan 0 jika tidak
3. Buatlah program untuk 1000 orang (laki-laki dan perempuan) yg akan dibuat data statistik jenis kelamin, bulan kelahiran, hari kelahiran dan yg terbanyak di lahir di bulan, hari dan jenis kelamin apa?? Untuk mendapatkan jenis kelamin, bulan dan hari lahir secara acak (Random). Output :

	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Ags	Sep	Okt	Nov	Des
Sen	7	7	9	4	3	10	7	9	5	7	6	4
Sel	6	6	4	5	7	5	5	6	4	7	2	4
Rab	9	5	9	5	4	6	4	3	14	8	7	7
Kam	6	2	5	9	7	1	11	6	0	3	7	7
Jum	1	7	10	1	2	8	7	7	1	7	8	4
Sab	3	7	11	5	7	9	6	7	8	3	6	6
Min	5	7	8	7	9	7	8	4	8	8	8	13

Jumlah Laki-laki 517

	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Ags	Sep	Okt	Nov	Des
Sen	11	5	2	4	6	9	4	3	7	8	5	7
Sel	6	6	10	1	4	4	5	4	10	6	6	7
Rab	6	6	3	6	9	7	6	5	13	3	3	6
Kam	9	7	4	7	4	4	11	1	7	6	5	5
Jum	8	5	6	4	7	5	9	5	4	1	4	2
Sab	3	4	1	6	10	8	9	6	7	8	7	8
Min	1	8	7	7	5	7	6	3	2	5	7	5

Jumlah Perempuan 483

Terbanyak adalah Laki-laki di bulan Sep hari Rab 14 Orang

4. Buat program untuk menampilkan operasi himpunan A dan B : Intersection (Irisan), Union(Gabungan) dan Difference (Selisih). Input himpunan A dan B dengan karakter 'A...Z' (maks. 7 char) dan lakukan validasi jika yg diinput bukan karakter 'A..Z'.

OPERASI HIMPUNAN	
=====	
Himpunan A	: { A,B,C,G,L,U,K}
Himpunan B	: { G,K,H,D,Q,R,A}
Intersection(A,B)	= {A,G,K}
Union (A, B)	= {A,B,C,D,G,H,K,L,Q,R,U}
Difference (A, B)	= {B,C,L,U}
Difference (B, A)	= {K,H,D,Q,R}

MODUL II

STACK

Tujuan Praktikum :

1. Mahasiswa dapat memahami konsep stack.
2. Mahasiswa dapat mengimplementasikan stack untuk memecahkan masalah tertentu.

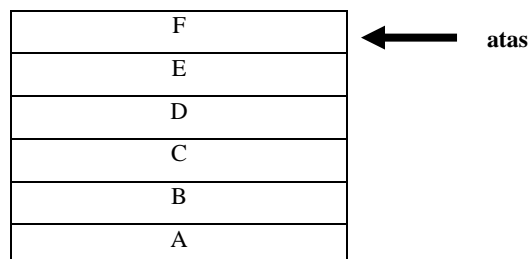
Tugas Pendahuluan :

- a. Buatlah deskripsi singkat tentang stack (dengan operasi-operasinya)!
- b. Buatlah algoritma untuk mengkonversikan infix menjadi postfix dan postfix menjadi infix!

Landasan Teori :

Secara sederhana stack atau tumpukan bias diartikan sebagai *kumpulan data yang seolah-olah diletakkan di atas data yang lain*. Satu hal yang perlu diingat bahwa kita bisa *menambah (menyisipkan) data dan mengambil (menghapus) data melalui ujung yang sama, yang disebut sebagai ujung atas stack*.

Secara sederhana, sebuah stack bisa diilustrasikan seperti gambar dibawah ini :



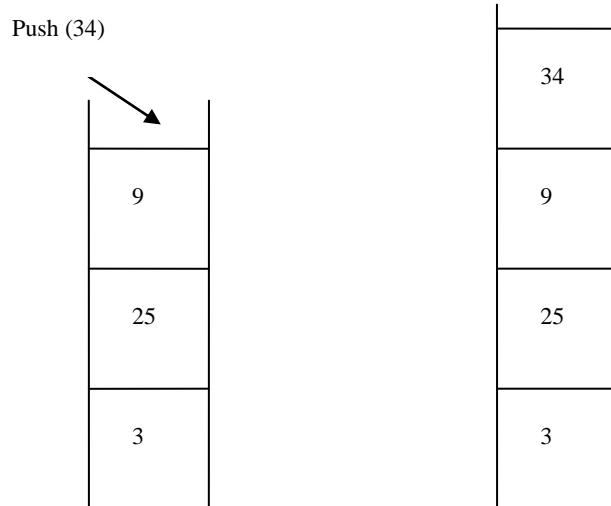
Dari gambar diatas, bisa di lihat bahwa kotak B terletak di atas kotak A dan di bawah kotak C. Kotak D terletak diatas kotak C, kotak E terletak diatas kotak D dan seterusnya sampai kotak terakhir. Dari gambar diatas menunjukkan bahwa kotak lewat satu ujung, yaitu bagian atas. Apabila ada kotak lain yang akan disisipkan, maka kotak tersebut akan diletakkan diatas kotak F, dan jika ada kotak yang akan diambil, maka kotak F yang pertama yang akan diambil.

Ada 2 operasi dasar yang bisa dilaksanakan pada sebuah stack, yaitu ***operasi menyisipkan data (push) dan operasi menghapus data (pop)***.

Operasi Push

Perintah *push* digunakan untuk memasukkan data ke dalam stack Untuk lebih jelasnya perhatikan ilustrasi berikut ini. Misalkan kita mempunyai data-data 3, 25 dan 9

dalam stack dengan posisi 3 paling bawah dan 9 paling atas. Dan kita akan memasukkan data 34 ke dalam stack tersebut. Tentu saja data 34 akan diletakkan diatas data 9.



Prosesnya dari operasi push dapat dilihat pada penggalan program berikut ini :

```

Procedure PUSH (var T : Tumpukan; X : integer);
begin
  if T.Atas = MaxElemen then
    writeln ('TUMPUKAN SUDAH PENUH')
  else
    begin
      T.Atas := T.Atas + 1;
      T.Isi[T.Atas] := X
    end
end;

```

Operasi Pop

Operasi pop adalah operasi untuk menghapus elemen yang terletak pada posisi paling atas dari sebuah tumpukan. Sama halnya dengan operasi push, maka dengan deklarasi tumpukan seperti diatas, prosedur untuk operasi pop bisa dengan mudah kita implementasikan, yaitu :

```

Procedure POP (var T : Tumpukan);
begin
  if T.Atas = 0 then
    writeln ('TUMPUKAN SUDAH KOSOSNG')
  else
    T.Atas := T.Atas - 1
end;

```

Pemanfaatan Stack

Salah satu pemanfaatan stack adalah untuk menulis ungkapan dengan menggunakan notasi tertentu. Seperti kita ketahui, dalam penulisan ungkapan numeris, kita selalu menggunakan tanda kurung untuk mengelompokkan bagian mana yang akan dikerjakan terlebih dahulu.

Sebagai contoh, perhatikan ungkapan berikut ini :

$$(C+D) *(E-F)$$

Dari contoh diatas $(C+D)$ akan dikerjakan lebih dahulu, kemudian baru $(E-F)$ dan hasilnya akan dikalikan.

Lain halnya dengan contoh berikut ini : $C+D * E-F$

$D * E$ akan dikerjakan terlebih dahulu, kemudian diikuti yang lain. Dalam hal ini pemakaian tanda kurung sangat penting karena akan mempengaruhi hasil akhir.

Cara penulisan ungkapan sering disebut dengan notasi infix, yang artinya bahwa operator ditulis diantara 2 operand. Bisa dirubah kedalam notasi prefix, yang artinya adalah bahwa operator ditulis sebelum kedua operand yang akan disajikan.

Perhatikan contoh dari notasi infix dan prefix berikut ini :

Infix	Prefix
$A+B$	$+AB$
$A+B-C$	$-+ABC$
$(A+B)*(C-D)$	$*+AB-CD$

Secara sederhana, proses konversi dari infix dapat dijelaskan sebagai berikut :

Misalkan ungkapan yang akan dikonversikan adalah sebagai berikut : $(A+B)*(C-D)$

Dengan menggunakan tanda kurung bantuan, ungkapan diatas dapat ditulis menjadi :

$$(+AB)*(-CD)$$

Jika $(+AB)$, kita umpamakan P dan $(-CD)$ sebagai Q, maka ungkapan diatas dapat ditulis : $P*Q$. Selanjutnya, notasi infix diatas diubah menjadi notasi prefix : $*PQ$. Dengan mengembalikan P dan Q pada notasinya semula dan menghapus tanda kurung bantuan, kita dapat memperoleh notasi prefix dari persamaan $(A+B)*(C-D)$, yaitu : $*+AB-CD$.

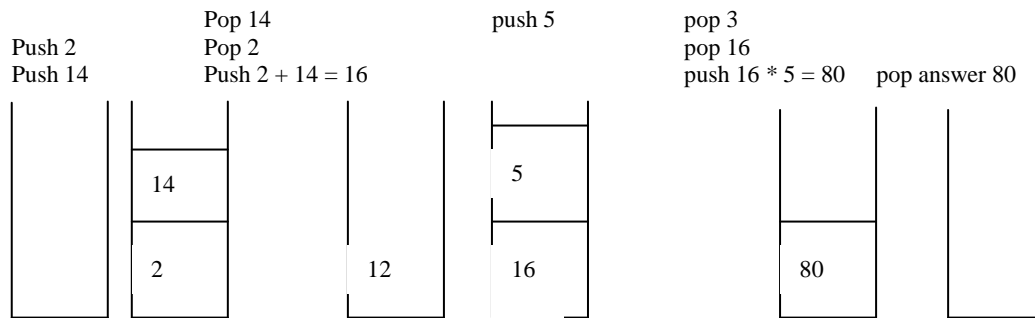
Notasi lain, yang merupakan kebalikan dari notasi prefix adalah notasi postfix. Dalam hal ini operator ditulis sesudah operand. Sama halnya dengan notasi prefix, maka notasi postfix dapat menggunakan tanda kurung bantuan.

Perhatikan contoh dari notasi infix dan postfix berikut ini :

Infix	Postfix
$16 / 2$	$16 2 /$
$(2 + 14) * 5$	$2 14 + 5 *$
$2 + 14 * 5$	$2 14 5 * +$
$(6 - 2) * (5 + 4)$	$6 2 - 5 4 + *$

Perhatikan ilustrasi yang menggambarkan penggunaan notasi postfix dalam stack.

Contohnya adalah $2 14 + 5 * = 80$



Tugas Praktikum :

(Asisten yang menentukan 2 soal saja yang dikerjakan oleh praktikan, bisa untuk masing-masing praktikan soalnya berbeda)

1. Buat program untuk menyiapkan array satu dimensi yang akan digunakan sebagai stack S sebanyak 20 elemen, bertipe integer. Kemudian lakukan proses simpan data ke stack atau mengeluarkan data dari stack dengan proses sebagai berikut :
 - a. Inputkan data dari keyboard
 - b. Bila data yang diinput bernilai 999, maka proses selesai
 - c. Bila data yang diinputkan bernilai ≥ 60 , maka periksa kondisi stack. Bila stack masih bisa diisi, maka simpan data tersebut/PUSH dan proses diulangi kembali mulai no. 1. Tapi bila stack sudah penuh, data tidak jadi disimpan dan cetak “Stack penuh” dan proses selesai
 - d. Bila data yang diinput < 60 , maka periksa kondisi stack. Bila stack ada isinya, ambil isi stack/POP dan proses kembali mulai no.1. Tapi bila stack tidak ada isinya, maka cetak “Stack kosong” dan proses selesai.
2. Memeriksa kelengkapan pasangan kurung buka dan kurung tutup pada suatu operasi aritmatika. Susun program untuk menginput sebuah operasi aritmatika (tidak lebih dari 20 karakter). Kemudian periksa kurung buka dan kurung tutup apakah tepat saling berpasangan. Cetak “BENAR” bila tepat saling berpasangan, atau cetak “SALAH” bila ada yang tidak tepat saling berpasangan

Misalnya :

$A + B * C - D / E$ benar

$A + B * (C - D) / E$ benar

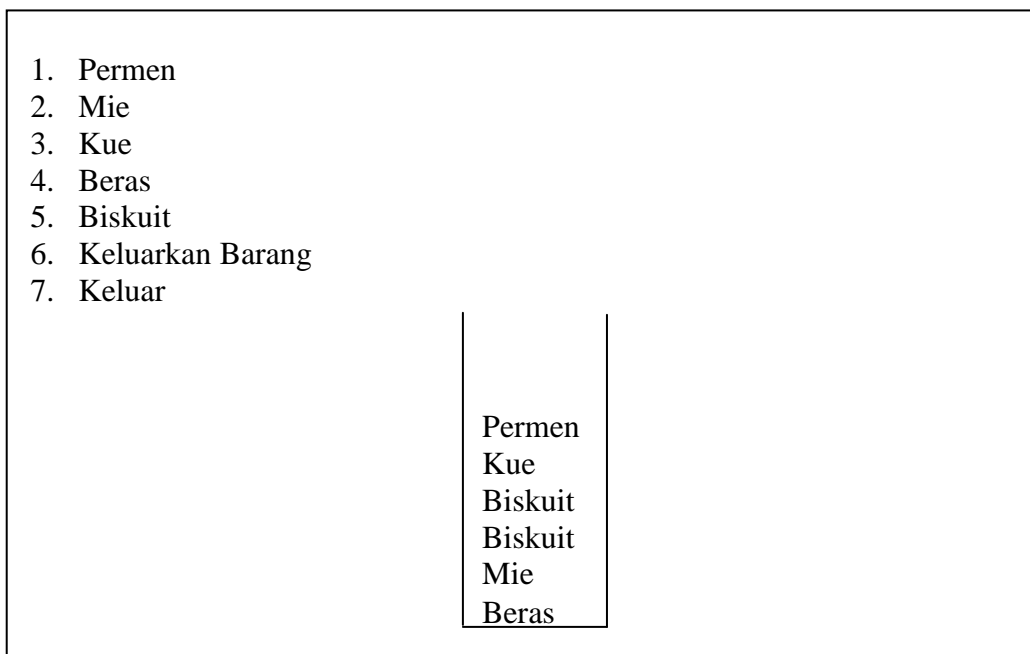
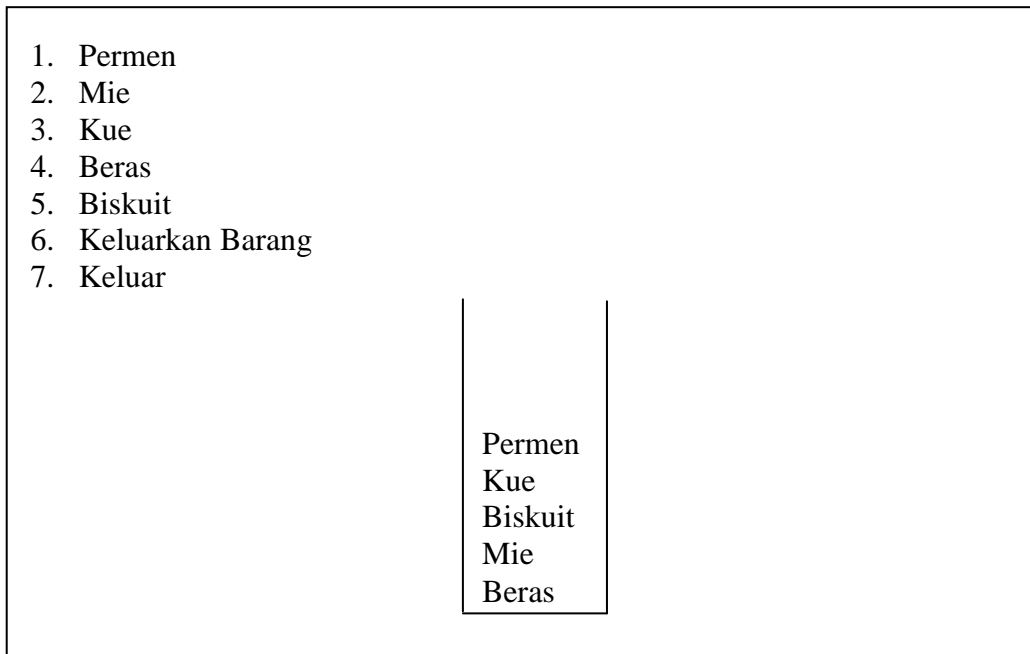
$(A + B) * (C - D) / E$ benar

$(A + B) * C - D) / E$ salah

$(A + B (* (C - D) / E$ salah

3. Buat program untuk mengubah notasi infix ke postfix dengan inputan data melalui keyboard.

4. Anda diminta bagian gudang untuk membuat program menyusun barang yang akan disimpan di gudang. Tentu saja harus diperhatikan bobot dari barang yang dimasukkan (posisi barang terberat harus berada paling bawah) . Barang-barang yang dimasukkan antara lain : Permen(berbobot 1), Kue dan Biskuit (Keduanya berbobot 2) ,Mie (berbobot 3), dan Beras (berbobot 4). Program ini juga menyediakan pula kemampuan mengeluarkan barang dari gudang tersebut.



Tampilan setelah dimasukkan biskuit lagi

MODUL III SEARCHING

Tujuan Praktikum :

1. Mahasiswa dapat memahami macam-macam algoritma pencarian.
2. Mahasiswa dapat menentukan algoritma pencarian yang paling cepat dan tepat untuk suatu masalah tertentu.

Tugas Pendahuluan :

- a. Buatlah algoritma pencarian data dengan menggunakan metode binary search dan sequential search !
- b. Buatlah algoritma pencarian data mahasiswa (NRP, nama mahasiswa, alamat, telpon) berdasarkan NRP !

Landasan Teori :

Definisi Searching

Pencarian Data (Searching) digunakan untuk mencari suatu data (bisa satu atau lebih) dari sejumlah data yang ada. Pembahasan tentang searching di sini dibagi menjadi 2, yaitu untuk data terurut dan data tidak terurut.

Data Tidak Terurut

Untuk mencari suatu data dari kumpulan data yang tidak terurut, cara yang digunakan adalah membandingkan satu persatu keseluruhan data yang ada dengan data yang dicari.

Contoh :

Terdapat N data dalam sebuah array A.

Data yang ingin dicari disimpan dalam variabel bilangan.

Kita ingin mencari isi dari bilangan dalam array A. Caranya sebagai berikut :

```
For a:= 1 to N do
  If A[i] = bilangan then
    Writeln ('Data',bilangan,'terdapat pada array ke :',i);
```

Data Terurut

Untuk data terurut, salah satu metode yang dapat digunakan adalah binary search.

Misalkan kite akan mencari data 20 dari array di bawah ini :

2 8 11 15 18 19 20 22 35 40 45

Caranya, array diatas dibagi menjadi 2 sub array :

2 8 11 15 18 19 20 22 35 40 45

Sub array 1

Sub array 2

Dari hasil pemecahan dapat kita lihat bahwa data yang bernilai 20 terdapat pada sub array 2. Dengan demikian pencarian dilaksanakan pada sub array 2, sub array 1 tidak perlu dihiraukan lagi. Sub array 2 dipecah lagi menjadi:

19 20 22 35 40 45

Sub array 1

Sub array 2

Sekarang data 20 terdapat pada sub array 1. Dengan demikian sub array 1 dipecah lagi. Proses diteruskan sampai data yang dicari ketemu atau tidak ketemu.

Pencarian Biner

Metode pencarian biner dijelaskan sebagai berikut. Setelah vektor yang diketahui diurutkan, vektor tersebut dibagi menjadi dua subvektor yang mempunyai jumlah elemen yang sama. Kemudian data dibandingkan dengan data terakhir dari subvektor pertama. Jika data yang dicari lebih kecil, pencarian diteruskan pada subvektor pertama dengan terlebih dahulu membagi dua subvektor tersebut. Tetapi jika data yang dicari lebih besar dari data terakhir pada subvektor pertama, berarti data yang dicari kemungkinan terletak pada subvektor kedua. Proses diatas diulang sampai data yang dicari ditemukan atau tidak ditemukan. Untuk lebih memperjelas penjelasan di atas perhatikan contoh berikut. Misalkan kita akan mencari data yang bernilai 20 pada vektor berikut ini.

2	8	11	15	18	19	20	22	35	40	45
---	---	----	----	----	----	----	----	----	----	----

Vektor di atas kita pecah menjadi 2 (dua) subvektor sebagai berikut :

2	8	11	15	18
---	---	----	----	----

Subvektor 1

19	20	22	35	40	45
----	----	----	----	----	----

Subvektor 2

Dari hasil pemecahan di atas kita lihat bahwa data yang bernilai 20 terdapat pada subvektor 2. Dengan demikian pencarian kita laksanakan pada subvektor 2, subvektor 1 tidak perlu dihiraukan lagi. Subvektor 2 kemudian dipecah lagi menjadi :

19	20	22
----	----	----

Subvektor 1

35	40	45
----	----	----

Subvektor 2

Sekarang, data yang bernilai 20 terdapat pada subvektor 1. Dengan demikian subvektor 1 kita pecah lagi. Proses diteruskan sampai data yang dicari ketemu atau tidak ketemu.

Tugas Praktikum :

(Asisten yang menentukan 2 soal saja yang dikerjakan oleh praktikan, bisa untuk masing-masing praktikan soalnya berbeda)

1. Buat program untuk menampilkan posisi suatu string dalam string yang lain, Contoh :

Posstr("Selamat", "amat")

Memberikan nilai balik berupa 3 yang menyatakan bahwa string "amat" terdapat pada string "Selamat" pada posisi ke-3 (posisi pertama dimulai dari 0). Jika String yang dicari tidak ditemukan, nilai balik berupa -1.

Pencarian data menggunakan sequential search.

2. Dengan cara yang pada soal no. 1 , terapkan dengan menggunakan binary search.
3. Terdapat array yang berisi n buah nama orang yang dimasukkan melalui keyboard. Buat program untuk menampilkan semua nama orang yang mengandung penggalan nama tertentu. Contoh :

Nama orang = "Budi Feriyanto", "Ahmad Budiyono", "Doni", "budi", "Fahmi
Rahmadan".

Dicari nama yang mengandung budi

Maka hasil :

Budi Feriyanto

Ahmad Budiyono

budi

4. Buatlah program untuk pencarian nilai rata-rata terbesar dari data mahasiswa yang terdiri : NPM, Nama, Mata Kuliah, Nilai Tugas, Nilai Quiz, Nilai UTS, Nilai UAS, dan nilai rata-rata diperoleh dari : $(10\% \times \text{Nilai Tugas}) + (10\% \times \text{Nilai Quiz}) + (30\% \times \text{Nilai UTS}) + (50\% \times \text{Nilai UAS})$.

MODUL IV

SORTING

Tujuan Praktikum :

1. Mahasiswa mampu memahami macam-macam algoritma pengurutan.
2. Mahasiswa mampu menentukan algoritma mana yang paling cepat untuk suatu masalah tertentu.

Tugas Pendahuluan :

- a. Buatlah algoritma pengurutan : bubble sort, quick sort, insertion sort dan selection sort !
- b. Buatlah algoritma memasukkan 10 nilai secara acak kemudian tampilkan output rangkaian 10 bilangan tersebut ascending / descending !

Landasan Teori :

Definisi Sorting

Sorting adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga menjadi tersusun secara teratur menurut suatu aturan tertentu.

Pada umumnya terdapat 2 jenis pengurutan :

- *Ascending (naik).*
- *Descending (turun).*

Contoh :

Data acak	:	5	6	8	1	3	25	10
Terurut Ascending	:	1	3	5	6	8	10	25
Terurut Descending	:	25	10	8	6	5	3	1

Untuk melakukan proses pengurutan tersebut dapat digunakan berbagai macam cara / metoda. Beberapa metoda diantaranya :

1. Bubble / Exchange Sort
2. Selection Sort
3. Insertion Sort
4. Quick Sort

Metode Bubble Sort

Metode bubble sort adalah metode yang mendasarkan penukaran dua buah elemen untuk mencapai keadaan urut. Metode ini adalah yang termudah, tetapi paling tidak efisien.

Iterasi ke	A[1]	A[2]	A[3]	A[4]	A[5]
Awal	24	23	56	45	12

I = 1	23	24	56	45	12
	23	24	56	45	12
	23	24	45	56	12
	23	24	45	12	56
I = 2	23	24	45	12	56
	23	24	45	12	56
	23	24	12	45	56
I = 3	23	24	12	45	56
	23	12	24	45	56
I = 4	12	23	24	45	56
Akhir	12	23	24	45	56

Algoritma bubble sort

[data yang akan diurutkan dimasukkan ke dalam sebuah array, dalam contoh ini, array tersebut diberi nama A[..], dimana isi dari array A belum terurut]

Langkah 0 *Baca array yang akan diurutkan*

Langkah 1 *Untuk I = 1 sampai N-1, kerjakan langkah 2*

Langkah 2 *Untuk J = 1 sampai N-I, kerjakan langkah 3*

Langkah 3 *Apakah A[J] > A[J+1] ?*

Jika ya, tukarkan nilai kedua elemen ini.

Langkah 4 *Selesai*

Metode Selection Sort

Pada langkah pertama, dicari data terkecil dari data pertama sampai data terakhir. Kemudian data terkecil tersebut ditukar dengan data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding dengan data yang lain. Pada langkah kedua, data terkecil kite cari mulai data kedua sampai data terakhir. Data terkecil yang diperoleh ditukar dengan data kedua. Demikian seterusnya sampai seluruh data terurut.

Iterasi ke	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
I=1, Lok=3	23	45	12	24	56	34	27	23	16
I=2, Lok=9	12	45	23	24	56	34	27	23	16
I=3, Lok=3	12	16	23	24	56	34	27	23	45
I=4, Lok=8	12	16	23	24	56	34	27	23	45
I=5, Lok=8	12	16	23	23	56	34	27	24	45
I=6, Lok=7	12	16	23	23	24	34	27	56	45
I=7, Lok=7	12	16	23	23	24	27	34	56	45
I=8, Lok=9	12	16	23	23	24	27	34	56	45
Akhir	12	16	23	23	24	27	34	45	56

Algoritma Selection Sort :

Input data dinyatakan dalam array A (belum terurut), dan banyak elemen adalah N.

Langkah 0 *Baca array yang akan diurutkan*

Langkah 1 *Untuk J = I + 1 sampai N, kerjakan langkah 2 sampai 4*

- Langkah 2** *Tentukan $Lok = I$*
 Untuk $J = J + 1$ sampai N , kerjakan langkah 3
- Langkah 3** *Apakah $A[Lok] > A[J]$*
 Jika ya, tentukan : $Lok J$
- Langkah 4** *Tukarkan nilai $A[Lok]$ dengan $A[J]$*
- Langkah 5** *Selesai*

Tugas Praktikum :

(Asisten yang menentukan 2 soal saja yang dikerjakan oleh praktikan, bisa untuk masing-masing praktikan soalnya berbeda)

1. Buat program untuk mengurutkan data dengan menggunakan bubble sort. Data yang akan diurutkan mempunyai empat data, yaitu :
 - NRP, bertipe bilangan bulat
 - Nama, bertipe string
 - Mata Kuliah, bertipe string
 - Nilai, bertipe bilangan real

Prosedur pengurutan menerima satu parameter, yaitu bilangan bulat yang dapat bernilai 1, 2, 3 atau 4. Apabila bernilai 1, maka data diurutkan menurut NRP. Apabila bernilai 2, maka data diurutkan menurut nama. Apabila bernilai 3, maka data diurutkan menurut mata kuliah. Apabila bernilai 4, maka data diurutkan berdasarkan nilai. Semua data tersebut diurutkan secara Ascending.

2. Modifikasi program no. 1 dengan menambahkan pencarian data berdasarkan NRP, nama, mata kuliah atau Nilai.
3. Buatlah program pengurutan dengan metode quick sort no. 1 dengan pengurutan secara ascending dan descending
4. Buat program yang bisa mengurutkan data yang dimasukkan melalui keyboard menggunakan metode selection sort. Program ini bisa mengurutkan data berdasarkan dua kunci atau lebih. Data yang dapat digunakan yaitu NomorInduk, Nama, Alamat, dan Golongan. Data ini dapat diurutkan menurut Nama + Alamat atau Golongan + Nama. Sebagai contoh, jika data diurutkan menurut Nama + Alamat, dan terdapat dua orang dengan nama sama, misalkan Deni dan Deni, maka dua data ini diurutkan menurut alamatnya. Jika Deni pertama beralamat di Surabaya dan Deni kedua beralamat di Bandung, maka Deni kedua harus diletakkan sebelum Deni pertama.
5. Buatlah algoritma untuk mengurutkan data-data berikut ini dengan menggunakan metode insertion sort, secara ascending dan descending.

Data – datanya : desa, deta, deden, denai, dasar, dudun, dusun

6. Buatlah program yang akan menampilkan sejumlah angka yang dipilih secara acak, disertai grafik batang yang mewakili besar masing-masing angka. Jika sembarang tombol ditekan maka angka-angka tersebut beserta grafik batangnya masing-masing akan diurutkan dari kecil ke besar dengan metode insertion sort.

```
#####20
#####8
#####13
#####10
#####25
```

Maka setelah diurutkan :

```
#####8
#####10
#####13
#####20
#####23
```

MODUL V

POINTER

Tujuan Praktikum:

1. Mahasiswa mampu memahami konsep pointer.
2. Mahasiswa mampu melakukan operasi – operasi pada pointer.
3. Mahasiswa mampu mengimplementasikan konsep pointer.

Tugas Pendahuluan :

- a. Buatlah deskripsi singkat tentang pointer !
- b. Buatlah algoritma penyajian pointer dalam program (deklarasi pointer) !
- c. Apa perbedaan antara Array dan Pointer? Jelaskan dan beri contoh!

Landasan Teori

Pointer adalah variabel yang berisi alamat memori sebagai nilainya dan berbeda dengan variabel biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variabel yang mempunyai nilai tertentu. Seperti halnya dengan tipe data yang lain, tipe pointer biasanya dideklarasikan pada bagian deklarasi tipe.

Bentuk umum deklarasi pointer adalah :

```
type pengenal = ^simpul;  
simpul = tipe;
```

dengan pengenal : nama pengenal yang menyatakan data bertipe pointer.

simpul : nama simpul.

tipe : tipe data dari simpul

Dalam kebanyakan program-program terapan, biasanya terdapat sekumpulan data yang dikumpulkan dalam sebuah rekaman (record), sehingga anda akan banyak menjumpai tipe data pointer yang elemennya (data yang ditunjuk) adalah sebuah rekaman. Perhatikan contoh berikut :

```
type Str30 = string[30];  
Point = ^Data;  
Data = record  
    Nama_Peg : Str30;  
    Alamat : Str30;  
    Pekerjaan : Str30;  
end;
```

Dengan deklarasi tipe data seperti diatas, kita bisa mendeklarasikan perubah, misalnya :

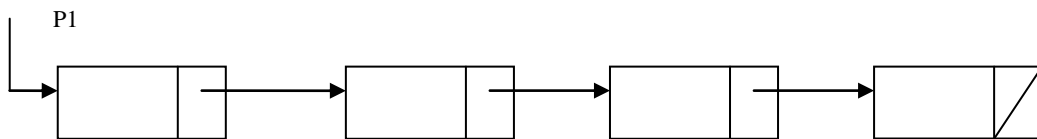
```
var P1, P2 : Point;  
    A, B, C : Str30;
```

Untuk mengalokasikan simpul dalam pengingat, statemen yang digunakan adalah statemen *new*, yang mempunyai bentuk umum :

new (perubah);

dengan perubah adalah nama perubah yang bertipe pointer.

Jika kita benar-benar ingin mempunyai suatu perubah yang benar-benar bersifat dinamis, maka kita harus mampu untuk memasup sejumlah lokasi tertentu dengan hanya menggunakan sebuah pointer awal, misal seperti pada gambar dibawah ini :



Untuk membentuk keadaan diatas, maka deklarasi tipe pointer kita ubah menjadi:

```

type perubah = ^simpul;
simpul = record
    info : tipe;
    berikut : perubah
end;

```

dengan perubah : nama perubah yang bertipe pointer.

simpul : nama simpul.

info : nama medan dari data yang bisa bertipe sembarang.

tipe : tipe data dari masing-masing medan.

berikut : nama medan yang bertipe data pointer.

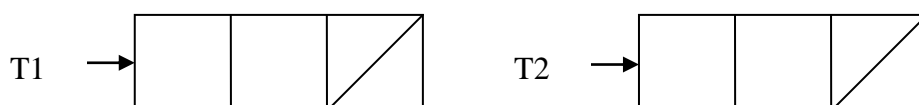
Operasi Pada Pointer

Operasi pada pointer ada 2 :

1. Mengkopi pointer, sehingga sebuah simpul akan ditunjuk oleh lebih dari sebuah pointer.
2. Mengkopi isi simpul, sehingga dua atau lebih simpul yang ditunjuk oleh pointer yang berbeda mempunyai isi yang sama.

Syarat yang harus dipenuhi untuk kedua operasi ini adalah bahwa pointer-pointer yang akan dioperasikan harus mempunyai deklarasi yang sama.

Kita mempunyai pointer T1 dan T2 yang mempunyai deklarasi simpul yang sama

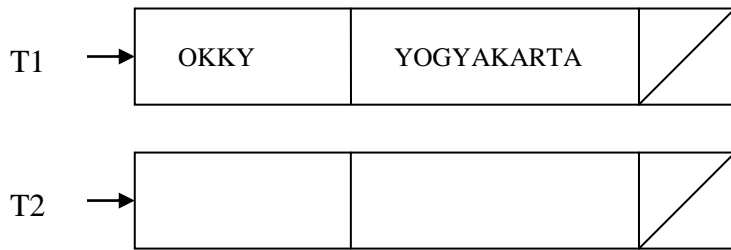


Dengan menggunakan statemen :

T1^.Nama := 'OKKY';

$T1^{\wedge}.Alamat := 'YOGYAKARTA';$

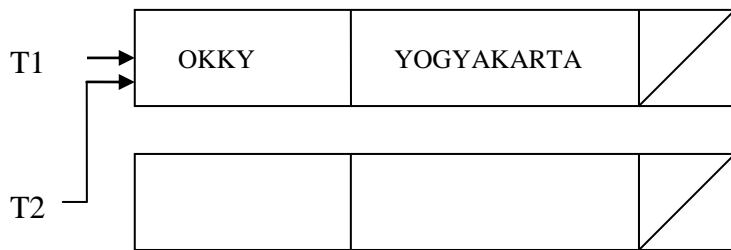
maka keadaan dua simpul diatas berubah menjadi :



Jika kita memberikan statemen :

$T2 := T1;$

Maka gambar diatas berubah menjadi :



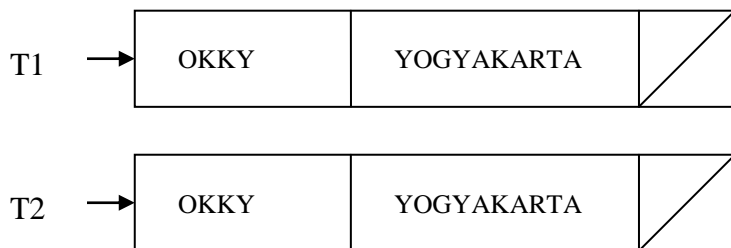
Operasi diatas yang disebut dengan operasi mengkopi pointer.

Jika statemen yang kita berikan adalah :

$T2^{\wedge} := T1^{\wedge};$

maka hasil yang kita peroleh adalah :

YOGYAKARTA



Operasi diatas yang disebut operasi mengkopi isi simpul.

Dari ilustrasi diatas, jelaslah apa yang dimaksud dengan operasi mengkopi pointer dan operasi mengkopi isi simpul. Sebagai ringkasan, maka :

- ✓ Jika dalam statemen pemberian tanda \wedge tidak ditulis, operasinya disebut operasi mengkopi pointer; dengan konsekuensi, simpul yang semula ditunjuk oleh suatu pointer akan bisa terlepas dan tidak dimasup lagi.

- ✓ Jika dalam statemen pemberian tanda ^ ditulis, operasinya disebut operasi mengkopi isi simpul pointer; dengan konsekuensi bahwa isi dua simpul atau lebih akan menjadi sama.

Menghapus Pointer

Di atas telah dijelaskan bahwa pointer yang telah dialokasikan (dibentuk) bisa didealokasikan (dihapus) kembali pada saat program dieksekusi. Setelah suatu pointer dihapus, maka lokasi yang semula ditempati oleh simpul yang ditunjuk oleh pointer tersebut akan bebas, sehingga bisa digunakan oleh perubah lain.

Statemen untuk menghapus pointer adalah ***dispose***, yang mempunyai bentuk umum :

dispose (perubah)

dengan perubah adalah sembarang perubah yang bertipe pointer.

Tugas Praktikum :

1. Buatlah program untuk membalik kata menggunakan pointer !
2. Buatlah program untuk mengurutkan data yang dimasukkan melalui keyboard yang disajikan dengan menggunakan pointer (metode bebas) !
3. Buat program untuk mengubah huruf dari kecil ke besar dan dari besar ke kecil?
4. Buat program untuk menghitung FPB (Faktor Persekutuan Terbesar) dari suatu bilangan yang diinputkan user?

MODUL VI

LINKED LIST

Tujuan Praktikum :

1. Mahasiswa mampu memahami konsep linked list.
2. Mahasiswa mampu melakukan operasi penambahan, penghapusan dan pembacaan pada linked list.
3. Mahasiswa mampu melakukan operasi penggabungan linked list.
4. Mahasiswa mampu mengimplementasikan konsep linked list.

Tugas Pendahuluan :

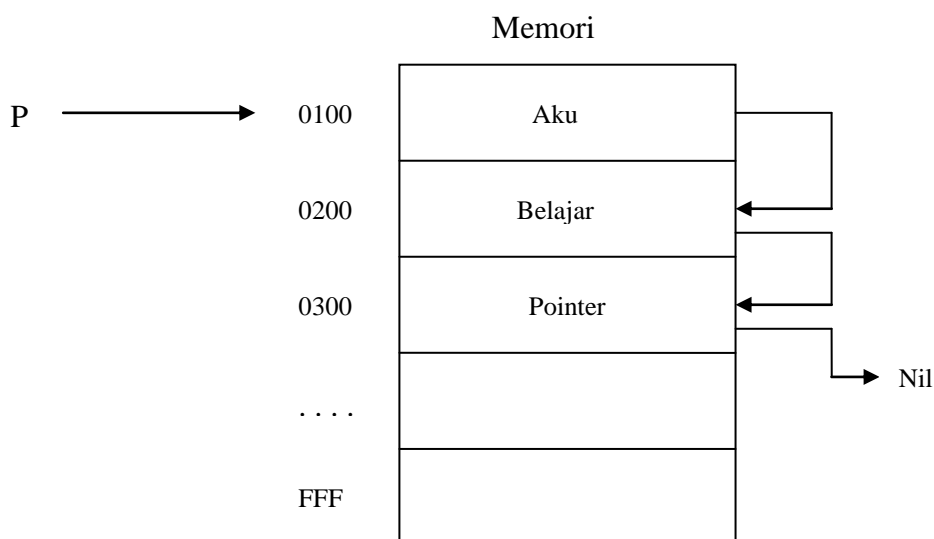
- a. Buatlah dekripsi singkat tentang link list !
- b. Buatlah algoritma daftar nama. Pertama tanyakan apa nama akan diurutkan secara FIFO (First In First Out) / LIFO (last In First Out). Lalu lakukan prosedur input yang meliputi nrp dan nama yang tadi diinput dengan urutan sesuai pilihan (LIFO/FIFO).

Landasan Teori :

Single Linked List

Apabila setiap kali kita ingin menambahkan data selalu dengan menggunakan variabel pointer yang baru, kita akan membutuhkan banyak sekali variabel pointer (penunjuk).

Oleh karena itu ada baiknya jika kita hanya menggunakan satu variabel pointer saja untuk menyimpan banyak data dengan metode yang kita sebut Linked List. Jika diterjemahkan, maka berarti suatu daftar isi yang saling berhubungan. Untuk lebih jelasnya perhatikan gambar dibawah ini :



Pada gambar di atas tampak bahwa sebuah data terletak pada sebuah lokasi memory area. tempat yang disediakan pada suatu area memory tertentu untuk menyimpan data dikenal dengan sebutan node/simpul. Pada setiap node memiliki pointer (penunjuk) yang menunjuk ke simpul berikutnya sehingga terbentuk suatu untaian dan dengan demikian hanya diperlukan sebuah variabel pointer. (ket : Nil tidak memiliki nilai apapun. Biasanya linked list pada titik akhirnya akan menunjuk ke Nil)

Dalam pembuatan Single Linked List dapat menggunakan 2 metode :

- LIFO (Last In First Out), aplikasinya : Stack (Tumpukan).
- FIFO (First In First Out), aplikasinya : Queue (Antrian).

LIFO

LIFO adalah suatu metode pembuatan Linked List dimana data yang masuk paling akhir adalah data yang keluar paling awal. Hal ini dapat dianalogikan dalam kehidupan sehari-hari pada saat menumpuk barang. Jika Linked List dibuat dengan metode LIFO, maka terjadi penambahan/insert simpul di belakang.

Prosedur Insert Pada LIFO

Istilah Insert berarti menambahkan sebuah simpul baru kedalam suatu linked list.

```

Type
Point      = ^RecPoint;
RecPoint   = Record
            Isi      : TipeData;
            Next    : Point;
            End;
Var
    Head, Tail, Now : Point;

```

Penggalan deklarasi tipe data dan variabel diatas akan dipakai pada penjelasan prosedur-prosedur selanjutnya.

```

Procedure INSERT (elemen : TipeData);
Var Now : Point;
begin
    New (Now);
    Now^.Isis := elemen;
    If Head = Nil then
        Now^.Next := Nil
    else
        Now^.Next := Head;
    Head := Now;
end;

```

FIFO

FIFO adalah suatu metode pembuatan Linked List dimana data yang masuk paling awal adalah data yang keluar paling awal juga. Hal ini dapat dianalogikan dalam

kehidupan sehari-hari misalnya saat kelompok orang yang datang mengantri hendak membeli tiket di loket. Jika linked list dibuat dengan metode FIFO, maka terjadi penambahan /insert simpul didepan.

Prosedur Insert Pada FIFO

```
Procedure INSERT (elemen : TipeData);
Var Now : Point;
begin
    New (Now);
    If Head = Nil then
        Head := Now
    else
        Tail^.Next := Now;
    Tail := Now;
    Tail^.Next := Nil;
    Now^.Isi := elemen;
end;
```

Prosedur Dan fungsi Linked List lainnya

Prosedur-prosedur serta fungsi umum dalam aplikasi linked list adalah :

- **Create** : Membuat sebuah linked list yang baru dan masih kosong. Prosedur ini wajib dilakukan sebelum menggunakan linked list.

```
Procedure Create;
begin
    Head := nil;
    Tail := nil;
end;
```

- **Empty** : Fungsi untuk menentukan apakah linked list kosong atau tidak

```
Function Empty : boolean;
begin
    If head = nil then
        Empty := true
    else
        Empty := false;
end;
```

- **Find First** : Mencari elemen pertama dari linked list.

```
Procedure Find_First;
begin
    Now := head;
end;
```

- **Find Next** : Mencari elemen sesudah elemen yang ditunjuk now.

```
Procedure Find_Next;
begin
    If Now^.next <> nil then
        Now := Now^.next;
end;
```

- **Retrieve** : Mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu ditampung pada suatu variabel, dicontohkan variabel r.

```

Procedure Retrieve(var r : TipeData);
  Begin
    r := Now^.isi;
  end;

```

- **Update** : Mengubah elemen yang ditunjuk oleh now dengan isi dari suatu variabel, dicontohkan variabel u.

```

Procedure Update(var u :TipeData);
  begin
    Now^.isi := u;
  end;

```

- **Delete Now** : Menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalah elemen pertama dari linked list (head), maka head akan berpindah ke elemen berikut.

```

Procedure DeleteNow;
  Var x : point;
  begin
    If Now <> head then
      begin
        x := head;
        While x^.next <> now do
          x := x^.next;
        x^.next := now^.next;
      end
    else head := head^.next;
    Dispose(now);
    Now := head;
  end;

```

- **Delete Head** : Menghapus elemen yang ditunjuk head. Head berpindah ke elemen sesudahnya.

```

Procedure DeleteHead;
  begin
    If head <> nil then
      begin
        Now := head;
        Head := head^.next;
        Dispose(now);
        Now := head;
      end;
    end;

```

- **Clear** : Untuk menghapus linked list yang sudah ada. Wajib dilakukan bila ingin mengakhiri program yang menggunakan linked list. Jika tidak, data-data yang dialokasikan ke memori pada program sebelumnya akan tetap tertinggal didalam memori.

```

Procedure Clear;

```

```

begin
  While head <> nil do
    begin
      Now := head;
      Head := head^.next;
      Dispose(now);
    end;
  end;
end;

```

Tugas Praktikum :

(Asisten yang menentukan 2 soal saja yang dikerjakan oleh praktikan, bisa untuk masing-masing praktikan soalnya berbeda)

1. Buatlah program untuk menginventaris buku yang ada pada sebuah perpustakaan. Bagi tiga jenis buku : Teknik, Ekonomi ,Hukum dan Pertanian. Mintalah user menginput data-data sebagai berikut : kode(dimulai dengan huruf T utk Teknik, E utk Ekonomi, P utk Pertanian dan H utk Hukum disertai dengan angka empat digit), Jenis(Teknik/Ekonomi/Pertanian/Hukum), judul serta jumlahnya. Sebelumnya user diminta input metode inventarisasi (FIFO/LIFO). Kemudian tampilkan seluruh Buku yang ada beserta data-datanya secara FIFO/LIFO seperti yang diminta user.
2. Modifikasi program no. 1 utk menghapus data buku apabila buku yang ada di perpustakaan ada yang rusak.
3. Modifikasi program no.1 dengan menambahkan pencarian data berdasarkan kode, jenis atau judul buku.
4. Buatlah program untuk mendeteksi password/kata sandi. Gunakan metode single linked list. Jika passwordnya benar, program akan selesai, jika salah maka user akan diminta memasukkan password kembali.
5. a. Operasi penambahan pada link list.
Buatlah suatu link list dengan data yang disimpan adalah karakter/huruf nama lengkap anda.
- b. Operasi penghapusan pada link list.
Dari barisan karakter nama anda hapuslah huruf ke 3,5,6 dari urutan link list tersebut.
- c. Operasi pembacaan pada link list.
Tampilkan rangkaian karakter dari nama yang telah dihapus tersebut.
6. Operasi penggabungan pada link list
Buatlah dua link list :
 - Link list dari rangkaian huruf nama lengkap anda
 - Link list dari nama lengkap ayah / ibu anda

Gabungkan dua link list tersebut untuk menghasilkan link list dari rangkaian nama anda dan nama ayah / ibu anda.

7. Buatlah program daftar mata kuliah. Pertama tanyakan apa mata kuliah akan diurutkan secara FIFO (First In First Out) / LIFO (last In First Out). Lalu lakukan prosedur input yang meliputi kode dan nama mata kuliah yang tadi diinput dengan urutan sesuai pilihan (LIFO/FIFO).