SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah : Struktur Data
Kode : TIS3213
Semester : III
Waktu : 2 x 3 x 50 Menit
Pertemuan : 4 & 5

A. Kompetensi

1. Utama

   Mahasiswa dapat memahami tentang konsep pemrograman menggunakan struktur tumpukan.

2. Pendukung

   Mahasiswa dapat mengetahui contoh pemanfaatan tumpukan.

B. Pokok Bahasan

Tumpukan

C. Sub Pokok Bahasan

- Pengertian Tumpukan
- Penyajian Tumpukan
- Operasi Pada Tumpukan
- Penulisan Ungkapan Numeris

D. Kegiatan Belajar Mengajar

Tahapan Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat Peraga
Pendahuluan	1. Menjelaskan perkuliahan yang akan dijalani dalam satu semester 2. Menjelaskan materi-materi perkuliahan dan buku-buku acuan yang akan dipergunakan dalam semester ini	Mendengarkan dan memberikan komentar	Notebook, LCD, Papan Tulis
Penyajian	1. Menjelaskan tentang pengertian tumpukan 2. Menjelaskan tentang cara penyajian	Memperhatikan, mencatat, dan memberikan	Notebook, LCD, Papan Tulis

	tumpukan dalam program 3. Menjelaskan tentang operasi-operasi pada tumpukan 4. Menjelaskan tentang cara penulisan ungkapan numeris	komentar. Mengajukan pertanyaan.	
Penutup	1. Mengajukan pertanyaan kepada mahasiswa. 2. Memberikan kesimpulan. 3. Mengingatkan akan kewajiban untuk pertemuan selanjutnya.	Memberikan komentar. Mengajukan dan menjawab pertanyaan	Notebook, LCD, Papan Tulis

E. Evaluasi

Evaluasi dilakukan dengan cara memberikan pertanyaan langsung dan tidak langsung kepada mahasiswa.

F. Daftar Referensi

1. P. Insap Santosa, *Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001*
2. Wirth Niklaus, "Algorithms and Data Structure", Prentice Hall Int. Inc, 1986
3. Antonie Pranata, *Algoritma dan Pemrograman*, J&J Learning Yogyakarta, 2000
4. Dwi Sanjaya, *Bertualang dengan Struktur Data di Planet Pascal*, J&J Learning Yogyakarta, 2001
5. Materi – Materi dari Internet.

RENCANA KEGIATAN BELAJAR MINGGUAN
(RKBM)

Mata Kuliah : Struktur Data
Kode : TIS3213
Semester : III
Waktu : 2 x 3 x 50 Menit
Pertemuan : 4 & 5

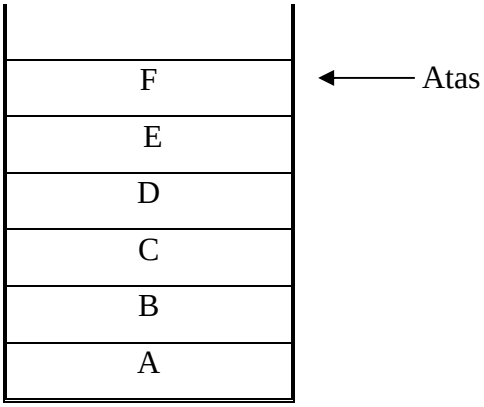
Minggu Ke-	Topik (Pokok Bahasan)	Metode Pembelajaran	Estimasi Waktu (Menit)	Media
1	2	3	4	5
4	3.1 Pengertian Tumpukan 3.2 Penyajian Tumpukan 3.3 Operasi Pada Tumpukan 3.3.1 Operasi Push 3.3.2 Operasi Pop	Ceramah, Diskusi Kelas	1 x 3 x 50’	Notebook, LCD, Papan Tulis
5	3.4 Penulisan Ungkapan Numeris	Ceramah, Diskusi Kelas	1 x 3 x 50’	Notebook, LCD, Papan Tulis

BAB III
TUMPUKAN

3.1 PENGERTIAN TUMPUKAN

Tumpukan (*stack*) bisa diartikan sebagai suatu kumpulan data yang seolah-olah ada data yang diletakkan diatas data yang lain. Dimana kita dapat menambah (menyisip) data dan mengambil (menghapus) data lewat ujung yang sama, yang disebut sebagai ujung atas tumpukan (*top of stack*).

Secara sederhana, sebuah tumpukan dapat diilustrasikan seperti Gambar 3.1 dibawah ini.



Gambar 3.1 Tumpukan 6 buah kotak

Dari gambar terlihat bahwa kotak F adalah bagian atas tumpukan, jika ada kotak lain yang akan disisipkan maka akan diletakkan diatas kotak F, dan jika ada kotak yang akan diambil maka kotak F yang akan diambil pertama kali.

Dengan memperhatikan ilustrasi diatas maka bisa dilihat bahwa tumpukan merupakan suatu senarai (*list*) yang mempunyai watak ”masuk terakhir keluar pertama” (*Last In First Out – LIFO*).

3.2 PENYAJIAN TUMPUKAN

Ada beberapa cara untuk menyajikan tumpukan dalam bahasa pemrograman. Cara paling sederhana adalah menggunakan larik (*array*). Dimana elemen tumpukan tidak melebihi maksimum elemen larik, karena jumlah elemen larik sudah tertentu (statis).

Cara lain, tumpukan bisa disajikan menggunakan rekaman (*record*). Dengan demikian, tumpukan bisa dideklarasikan sebagai berikut :

```
const MaxElemen = 255;
type Tumpukan = record
                    Isi : array [ 1 . . MaxElemen ] of integer;
                    Atas : 0 . . MaxElemen
                end;
var T : Tumpukan;
```

Dengan deklarasi diatas dianggap bahwa elemen tumpukan T, yang tersimpan dalam larik T.Isi, adalah bertipe integer dan banyaknya elemen tumpukan maksimum adalah sebesar MaxElemen, dalam hal ini 255 elemen.

3.3 OPERASI PADA TUMPUKAN

Ada dua operasi dasar yang bisa dilakukan pada sebuah tumpukan, yaitu operasi menyisipkan data (*push*), operasi menghapus data (*pop*). Karena kedalam tumpukan bisa mempush data, maka tumpukan sering disebut dengan *pushdown list*.

3.3.1 Operasi Push

Operasi Push adalah operasi untuk menambah (menyisip) elemen pada sebuah tumpukan. Prosedur untuk operasi push adalah sebagai berikut :

```
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;
```

3.3.2 Operasi Pop

Operasi pop adalah operasi untuk menghapus elemen yang terletak pada posisi paling atas dari sebuah tumpukan. Prosedur untuk operasi pop adalah sebagai berikut :

```
procedure POP (var T : Tumpukan);
begin
   if  T.Atas = 0 then
```

```
        writeln ('TUMPUKAN SUDAH KOSONG')
    else
        T.Atas := T.Atas – 1
end;
```

3.4 PENULISAN UNGKAPAN NUMERIS

Salah satu pemanfaatan tumpukan adalah untuk menulis ungkapan menggunakan notasi tertentu. Dalam penulisan ungkapan khususnya ungkapan numeris, kita selalu menggunakan tanda kurung untuk mengelompokkan bagian mana yang harus dikerjakan lebih dulu.

Contoh :

1. (A + B) * (C – D)

   Suku (A + B) akan dikerjakan lebih dahulu, kemudian suku (C – D) dan terakhir mengalikan hasil yang diperoleh dari dua suku ini.
2. A + B * C – D

   Maka B * C akan dikerjakan lebih dahulu, diikuti yang lain.

   Dalam hal ini pemakaian tanda kurung akan sangat mempengaruhi hasil akhir.

Cara penulisan ungkapan sering disebut dengan **Notasi Infix**, yang artinya adalah operator ditulis diantara dua operand. Seorang ahli matematika yang bernama *Jan Lukasiewicz* kemudian mengembangkan satu cara penulisan ungkapan numeris yang disebut Notasi Polish atau **Notasi Prefix**, yang artinya operator ditulis sebelum kedua operand yang akan disajikan. Berikut disajikan beberapa contoh notasi prefix dari notasi infix.

Infix	**Prefix**
A + B	+ A B
(A + B) * (C – D)	* + A B – C D

Secara sederhana, proses konversi dari infix menjadi prefix adalah sebagai berikut :

1. Ungkapan yang akan dikonversikan adalah (A + B) * (C – D)
2. Dengan menggunakan tanda kurung bantuan, ungkapan diatas dirubah menjadi:

   [+ A B] * [- C D]

3. Jika [+ A B] dimisalkan P, dan [- C D] dimisalkan Q, maka ungkapan diatas bisa ditulis menjadi : P * Q
4. Selanjutnya notasi infix diatas dirubah menjadi notasi prefix : * P Q
5. Dengan mengembalikan P dan Q pada notasi semula dan menghapus tanda kurung bantuan, diperoleh notasi prefix dari persamaan (A + B) * (C – D) yaitu: * + A B – C D

Dalam hal ini urutan penulisan operator akan menentukan operasi mana yang harus dikerjakan lebih dahulu.

Berikut adalah algoritma untuk mengubah Notasi Infix menjadi Notasi Prefix :

Algoritma INFIX KE PREFIX

Langkah 0 :

- ☞ Baca ungkapan dalam notasi infix, misalnya S;
- ☞ Tentukan panjang ungkapan tersebut, misalnya N;
- ☞ Siapkan sebuah tumpukan kosong;
- ☞ Siapkan derajat masing-masing operator, misalnya : $ berderajat 3, * dan / berderajat 2, + dan – berderajat 1 dan ")" berderajat 0.

Langkah 1 :

Dimulai dari I = 1 sampai N, kerjakan langkah-langkah berikut :

a. R = S(I)

b. Test Nilai R, jika R adalah :

- ☞ Operand : Langsung ditulis
- ☞ Kurung Buka : *Pop* dan tulis semua isi tumpukan sampai ")", *pop* juga tanda ini tetapi tidak usah ditulis
- ☞ Kurung Tutup : *Push* kedalam tumpukan
- ☞ Operator : Jika tumpukan kosong, atau derajat R lebih tinggi dibanding derajat ujung tumpukan, *Push* operator kedalam tumpukan. Jika tidak *pop* ujung tumpukan dan tulis. Kemudian ulangi perbandingan R dengan ujung tumpukan, lalu R di *Push.*

Catt : Kurung tutup didalam tumpukan dianggap mempunyai derajat yang lebih rendah dibanding R.

Langkah 2 : Jika akhir notasi infix telah tercapai dan tumpukan masih belum kosong, *pop* semua isi tumpukan dan tulis hasilnya.

Notasi lain yang merupakan kebalikan notasi prefix adalah **Notasi Postfix** atau notasi suffix, atau lebih dikenal dengan Notasi Polish Terbalik (*Reverse Polish Notation – RPN*). Dalam hal ini operator ditulis sesudah operand. Sama halnya dengan notasi prefix, disini juga diperlukan tanda kurung pengelompokan.

Proses konversi dari notasi infix ke notasi postfix adalah :

1. Ungkapan yang akan dikonversikan adalah (A + B) * (C – D)
2. Dengan menggunakan tanda kurung bantuan, ungkapan diatas dirubah menjadi : [A B +] * [C D -]
3. Jika [A B +] dimisalkan P, dan [C D -] dimisalkan Q, maka ungkapan diatas bisa ditulis menjadi : P * Q
4. Selanjutnya notasi infix diatas dirubah menjadi notasi postfix : P Q *
5. Dengan mengembalikan P dan Q pada notasi semula dan menghapus tanda kurung bantuan, diperoleh notasi prefix dari persamaan (A + B) * (C – D) yaitu: A B + C D - *

Dalam hal inipun urutan penulisan operator juga menentukan operasi mana yang harus dikerjakan lebih dahulu.

Berikut disajikan beberapa contoh konversi notasi infix menjadi notasi postfix :

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

Berikut adalah algoritma untuk mengubah Notasi Infix menjadi Notasi Postfix :

Algoritma INFIX KE POSTFIX

Langkah 0 :

- ☞ Baca ungkapan dalam notasi infix, misalnya S;
- ☞ Tentukan panjang ungkapan tersebut, misalnya N;
- ☞ Siapkan sebuah tumpukan kosong;

- ☞ Siapkan derajat masing-masing operator, misalnya : $ berderajat 3, * dan / berderajat 2, + dan – berderajat 1 dan ”(” berderajat 0.

Langkah 1 :

Dimulai dari I = 1 sampai N, kerjakan langkah-langkah berikut :

a. R = S(I)

b. Test Nilai R, jika R adalah :

- ☞ Operand : Langsung ditulis
- ☞ Kurung Buka : *Push* kedalam tumpukan
- ☞ Kurung Tutup : *Pop* dan tulis semua isi tumpukan sampai “(“, *pop* juga tanda ini tetapi tidak usah ditulis
- ☞ Operator : Jika tumpukan kosong, atau derajat R lebih tinggi dibanding derajat ujung tumpukan, *Push* operator kedalam tumpukan. Jika tidak *pop* ujung tumpukan dan tulis. Kemudian ulangi perbandingan R dengan ujung tumpukan, lalu R di *Push.*

Catt : Kurung buka didalam tumpukan dianggap mempunyai derajat yang lebih rendah dibanding R.

Langkah 2 : Jika akhir notasi infix telah tercapai dan tumpukan masih belum kosong, *pop* semua isi tumpukan dan tulis hasilnya.

--ooOOOoo--

Soal & Pembahasan :

Soal :

1. Apakah yang dimaksud dengan tumpukan (*stack*).
2. Sebutkan operasi pada tumpukan.

Pembahasan :

1. Tumpukan (*stack*) adalah suatu kumpulan data yang seolah-olah ada data yang diletakkan diatas data yang lain.
2. Operasi pada tumpukan :
    - Operasi **Push** : operasi untuk menambah (menyisip) elemen pada sebuah tumpukan.
    - Operasi **Pop** : operasi untuk menghapus elemen yang terletak pada posisi paling atas dari sebuah tumpukan.