BAB 7

KOMBINASI STRUKTUR ALGORITMA PEMROGRAMAN

BAB 7
KOMBINASI STRUKTUR ALGORITMA PEMROGRAMAN

Capaian Pembelajaran

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu menganalisis, menyampaikan pendapat, dan mengoperasikan kombinasi struktur algoritma pemrograman pada sebuah kasus operasional bisnis.

Pokok Bahasan

1. Mengenali alur percabangan dalam perulangan (arsip beruntun)
2. Melakukan implementasi kombinasi percabangan dalam perulangan
3. Mengenali alur perulangan dalam percabangan (rekursif)
4. Melakukan implementasi kombinasi perulangan dalam percabangan

Evaluasi Pembelajaran

Soal Latihan Kombinasi Struktur Algoritma Pemrograman

Pre Test
Kombinasi Struktur Algoritma Pemrograman

1. Apa yang kamu ketahui tentang arsip beruntun?
2. Bagaimana contoh implementasi dari arsip beruntun?
3. Apa yang kamu ketahui tentang rekursif?
4. Bagaimana contoh implementasi dari rekursif?

Bab ini akan membahas terkait alasan mengapa perlu mempelajari dan memahami bagaimana kombinasi struktur algoritma pemrograman berperan penting dalam penguasaan keilmuan pemrograman dasar yang nantinya dapat digunakan untuk mempermudah pekerjaan terkait dengan pemrograman. Pemaparan dari bab ini bertujuan untuk memberikan pemahaman lebih lanjut mengenai lanjutan dari bab matriks, percabangan dan perulangan yang telah kita pelajari sebelumnya.

Memahami pentingnya kombinasi struktur algoritma pemrograman akan mendorong mahasiswa untuk belajar dan menerapkan algoritma-algoritma yang sudah dipelajari untuk diterapkan ketika praktik pemrograman secara riil. Di bagian akhir pada bab ini juga ada bahasan terkait latihan soal dan pembahasannya supaya mahasiswa dapat semakin paham mengenai kombinasi struktur algoritma pemrograman.

7.1 Mengenali Alur Percabangan dalam Perulangan

Sebelumnya sudah dijelaskan bahwa definisi algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah. Kemudian muncul notasi algoritma yang digunakan untuk memecahkan suatu masalah yang ada pada proses komputer menggunakan bermacam-macam teknik maupun metode. Inti dari notasi logaritma sendiri bukanlah semakin kompleks semakin bagus, namun kemudahan untuk membaca dan memahami notasi tersebut. Kemudian ada macam-macam struktur algoritma, yaitu struktur pemrograman sekuensial, struktur pemrograman percabangan, dan struktur pemrograman perulangan.

Di sini kita akan mempelajari terkait dengan kombinasi struktur algoritma pemrograman yang merupakan lanjutan dari bahasan struktur algoritma pemrograman. Kombinasi struktur algoritma pemrograman meliputi bagaimana kita dapat mengenali alur percabangan dalam perulangan beserta implementasinya.

Pada struktur pemilihan atau percabangan, terdapat pemeriksaan kondisi/syarat yang harus dipenuhi, yang kemudian akan memilih perintah apa yang akan dilakukan jika syarat tersebut dipenuhi. Perintah tidak lagi dikerjakan secara beruntun seperti pada struktur runtunan, tetapi berdasarkan syarat yang harus dipenuhi (Uce Indahyanti et al, 2020).

Untuk mengenali alur percabangan dalam perulangan kita perlu mempelajari penerapannya melalui pengarsipan beserta turunan-turunannya. Selain itu, untuk mempelajari lebih dalam kita juga perlu memahami dalam penerapan atau implementasinya. Definisi arsip (file) atau berkas adalah struktur penyimpanan data di dalam memori sekunder seperti disk. Data disimpan di dalam arsip agar sewaktu-waktu dapat dibuka kembali. Struktur arsip memungkinkan kita menyimpan data secara permanen dan mengaksesnya kembali bila perlu. Banyak aplikasi yang membutuhkan kemampuan ini. Pada umumnya, arsip menyimpan Informasi dari kategori yang sama. Misalnya, data karyawan disimpan di dalam arsip karyawan, data properti barang disimpan di dalam arsip properti, data keuangan disimpan di dalam arsip keuangan, dan sebagainya. Setiap arsip dikenali melalui namanya (Rinaldi Munir, 2007).

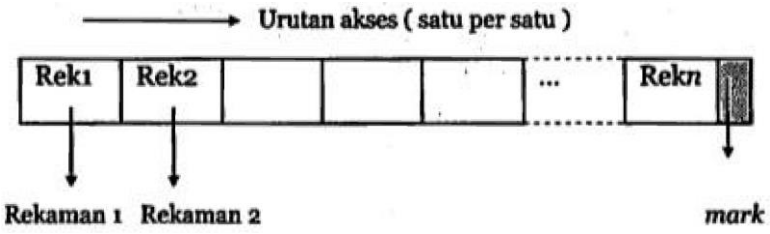
Setiap item data yang direkam di dalam arsip disebut rekaman (record). Semua rekaman di dalam arsip diorganisasikan penyimpanannya, dan pengaksesan rekaman di dalam arsip bergantung kepada metode pengorganisasiannya itu. Ada dua cara pengorganisasian data di dalam arsip: beruntun (sequential) dan acak (random) yang terakhir ini juga dikenal sebagai arsip akses langsung (direct access). Arsip beruntun (sequential file) menyimpan rekaman secara berurutan, rekaman yang satu sesudah rekaman yang lain. Untuk mengakses data tertentu, program harus mengakses mulai dari rekaman pertama sampai rekaman yang diinginkan. Pengaksesan arsip jenis ini pada umumnya sangat lambat, khususnya bila arsip berukuran besar. Namun, arsip beruntun mudah dibuat dan dipelihara (maintain).

Pada arsip acak, rekaman diakses secara langsung, tanpa perlu memulai dari awal. Keuntungan jenis arsip ini, pengaksesan data individual lebih cepat daripada arsip beruntun, tetapi arsip acak lebih sulit dibuat dan dipelihara. Dari kedua cara pengorganisasian itu, pengorganisasian rekaman secara beruntun adalah metode yang paling sederhana

7.2 Definisi Arsip Beruntun

Sub Program dengan jenis prosedur merupakan bagian dari kode program yang terpisah yang melakukan tugas spesifik dimana hasilnya dapat diketahui setelah semua proses didalamnya selesai dikerjakan. Prosedur perlu dipanggil dalam program utama agar hasil akhir dari sebuah prosedur dapat diketahui. Prosedur dapat dipanggil secara berulang kali di program utama. Sebagai contoh, prosedur untuk mengurutkan nilai dari IPK dari tiga Mahasiswa mulai yang terbesar ke terkecil. Sebelum prosedur dijalankan nilai IPK Mahasiswa belum terurut, namun setekah prosedur dijalankan, maka nilai IPK dari tiga mahasiswa sudah terurut dari terbesar ke terkecil.

Arsip beruntun adalah sekumpulan rekaman bertipe sama yang diakses secara berurutan mulai dari rekaman pertama sampai rekaman yang dituju atau sampai dengan rekaman yang terakhir. Rekaman di dalam arsip beruntun diakses satu per satu, yaitu rekaman demi rekaman, secara searah. Gambar 7.1 memperlihatkan gambaran lojik pengaksesan arsip beruntun. Setiap kotak pada gambar tersebut menyatakan sebuah rekaman. Kita dapat menganalogikan arsip beruntun seperti kumpulan lagu di dalam pita kaset. Untuk memainkan lagu pada urutan ke-6, kita harus memutar pita kaset mulai dari awal pita sampai awal lagu yang ke-6 ditemukan.



Gambar 7. 1 Struktur arsip beruntun

Perhatikanlah bahwa struktur arsip beruntuh tidak jauh berbeda dengan struktur larik. Perbedaannya, elemen larik didefinisikan di dalam memori, sedangkan elemen arsip didefinisikan di dalam media penyimpanan sekunder. Skema pemrosesan pada larik pada hakikatnya sama dengan skema pemrosesan pada arsip. Anda dapat mellhat bahwa beberapa algoritma traversal pada larik. seperti algoritma mencari nilai maksimum/minimum dan algoritma pencarian elemen tertentu, juga dapat digunakan pada arsip. Perbedaan arsip dengan larik hanya pada pillhan cara pengaksesan elemen data dan "arah" pemrosesan. Rekaman di dalam arsip beruntun tidak dapat dlakses secara langsung karena harus dlbaca dari awal rekaman, sementara pada larik elemennya dapat diakses secara langsung melalui indeksnya, Jika pada arsip beruntun pemrosesan rekaman hanya dapat dilakukan dalam satu arah, dimulai dari rekaman pertama sampai rekaman terakhir tetapi tidak dapat dari arah sebaliknya, maka pada larik kita dapat melakukan pemrosesan dari elemen terakhir menuju elemen pertama.

Jika pada larik kita dapat mengakhiri pemrosesan beruntun bilamana pencatat indeks larik sudah melebihi ukuran larik (biasa disimbolkan dengan N), maka pada arsip pembacaan rekaman berakhir jlka sudah sampai pada tanda (mark) yang menandakan akhir arsip (end of file). Tanda akhir arsip biasanya karakter khusus yang bergantung pada bahasa pemrograman yang digunakan. Kita mengasumsikan bahwa fungsi pustaka yang dapat mendeteksi akhir arsip sudah tersedia sehingga kita hanya memfokuskan pada algoritma pemrosesan arsip saja. Pemrograman tidak perlu tahu bagaimana cara fungsi tersebut mendeteksi akhir arsip.

7.3 Pendeklarasian Arsip Beruntun

Sebelum melakukan pemrosesan arsip beruntun, arsip tersebut harus dideklarasikan terlebih dahulu. Cara mendeklarasikan arsip beruntun di dalam bagian DEKLARASI sebagai berikut (sebagai peubah):

```
Deklarasi
arsip : File of tipe rekaman
```

Tipe rekaman dapal berupa tipe dasar (integer, real, char, boolean, string) atau tipe terstruktur (record). Setiap rekaman di dalam arsip beruntun harus bertipe sama, baik dari tipe dasar maupun bertipe terstruktur. Rekaman bertipe terstruktur terdiri atas satu atau lebih field yang bertipe tertentu.

```
Deklarasi
type nama tipe arsip : File of tipe rekaman { tipe bentukan }
arsip : nama tipe arsip
```

Berikut ini contoh-contoh pendeklarasian arsip beruntun di dalam bagian DEKLARASI:

1. Arsip Bil yang berisi sekurnpulan bilnngan bulat, Setiap rekaman adalah sebuah bilangan bulat.

```
Deklarasi
Bil : File of integer
```

2. Arsip Mhs yang berisi data mahasiswa (NIM, Nama, dan IP). Keterangan: NIM = Nomor Induk Mahasiswa, IP = Indeks Prestasi (skala 0 - 4). Setiap rekaman di dalam arsip Mhs bertipe terstruktur (record).

```
Deklarasi
{ tipe rekaman }
type DataMhs : record <NIM : integer, Nama : string, IP : real>
{ arsip }
Mhs : File of DataMhs
```

3. Arsip Kar, yaitu arsip yang berisi data bertipe karakter. Setiap rekaman adalah satu karakter. Perhatikan bahwa arsip karakter tidak sama dengan teks (text).

```
Deklarasi
Kar : File of char
```

4. Mendefinisikan arsip integer sebagai tipe bentukan

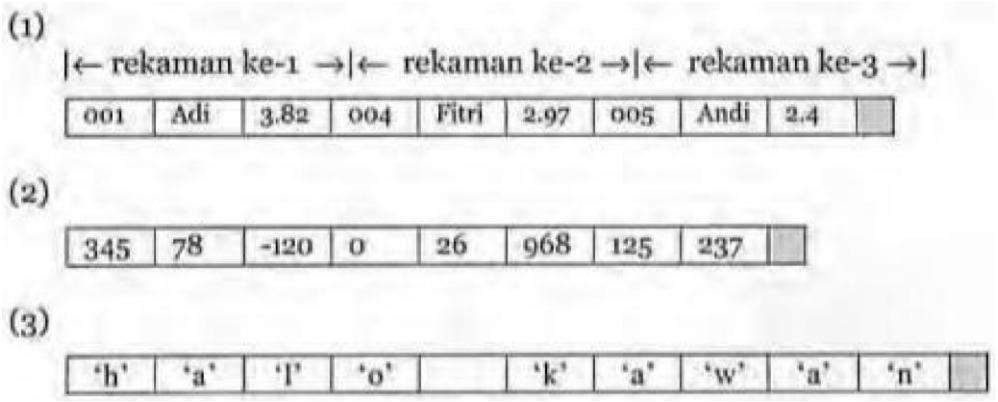
```
Deklarasi
type ArsipInt : File of integer { tipe bentukan }
Bil : ArsipInt { arsip bertipe ArsipInt }
```

5. Mendefinisikan arsip data mahasiswa pada contoh 2 di atas sebagai tipe bentukan.

```
Deklarasi
{ tipe rekaman }
type DataMhs : record <NIM : integer, Nama : string, IP : real>
```

```
{ tipe arsip }
type ArsipMhs : File of DataMhs
Mhs : ArsipMhs { arsip bertipe ArsipMhs}
```

Gambar 7.2 memperlihatkan contoh tiga buah arsip yang disebutkan di atas, berturut-turut (1) arsip data mahasiswa (Mhs), (2) arsip bilangan bulat (Bil), dan (3) arsip karakter (Kar). Garis-garis yang membatasi antara satu rekaman dengun rekaman lainnya hanyalah garis fiktif saja dan garis tersebut tidak terdapat dalam implementasi fisik arsip beruntun pada media penyimpanan sekunder. Rekaman yang diarsir menyatakan mark. Suatu arsip kosong adalah arsip yang hanya berisi mark.



Gambar 7. 2 Contoh tiga buah arsip

Contoh-contoh pendeklarasian arsip beruntun lninnya:

1. Mendeklarasian arsip Freal sebagai arsip yang berisi data bilangan riil.

```
Deklarasi
Freal : File of real
```

2. Mendeklarasikan ArsipTitik yang berisi titik-titik (x, y) di bidang kartesian.

```
Deklarasi
Type Titik : record <x : real, y : real>
ArsipTitik : File of Titik
```

7.4 Implementasi Kombinasi Percabangan dalam Perulangan (Contoh implementasi Arsip Beruntun)

Untuk memahami implementasi kombinasi perabangan dalam perulangan kita perlu mempelajari penerapannya melalui contoh-contoh implementasi arsip beruntun. Arsip hanya dapat diproses jika sudah terdefinisi isinya. Langkah pertama dalam membuat arsip adalah menyiapkan arsip untuk perencanaan (menggunakan perintah Open dengan Kode = 2). Langkah kedua adalah membaca data yang akan direkam (misalnya pembacaan dari piranti masukan), barulah kemudian menuliskan data tersebut ke dalam arsip (perintah FWrite). Jika sebuah arsip sudah selesai diisi, arsip tersebut ditutup dengan perintah Close.

Contoh 1. Mengisi arsip bilangan bulat dengan nilai 1 sampai n, Nilai n dibaca dari piranti masukan. Tanda akhir arsip didefinisikan 9999.

```
Deklarasi
Bil : File of integer { arsip bilangan bulat }
n : integer             { banyaknya bilangan bulat }
i : integer             { pencacah pengulangan }

ALGORITMA:
Open(Bil, 2)      {buka arsip Bil untuk perekaman }
read(n)
for i ← 1 to n do
      Fwrite(Bil, i)
endfor
Close(Bil)  { tutup arsip }
```

Contoh 2. Mengisi arsip bilangan bulat dengan nilai yang dibaca dari papan ketik. Tanda akhir arsip adalah 9999.

```
PROGRAM BuatArsipBilanganBulat2
{ Contoh program yang memperagakan cara menyimpan data ke dalam arsip, data
dibaca dari papan ketik. }

DEKLARASI
Fint : File of integer { arsip bilangan bulat }
n : integer             { banyaknya bilangan bulat }
i : integer             { pencacah pengulangan }
x : integer             { peubah bilangan bulat yang dibaca dari piranti
masukan }

ALGORITMA:
Open(Fint, 2)     { buka arsip Fint untuk perekaman }
read(n)
for i ← 1 to n do
      read(x)
```

```
Fwrite(Fint, x)
endfor
```

Contoh 3. Mengisi arsip Mhs dengan data mahasiswa yang dibaca dari papan ketik. Pembacaan data diakhiri bila NIM yang dimasukkan udalah 9999.. Mengisi arsip bilangan bulat dengan nilai 1 sampai n, Nilai n dibaca dari piranti masukan. Tanda akhir arsip didefinisikan 9999.

```
PROGRAM BuatArsipMahasiswa
{ Membuat arsip data mahasiswa, Data dibaca dari papan ketik. }

DEKLARASI
type DataMhs : record <NIM : integer, Nama : string, IP : real>
Msiswa : DataMhs { peubah untuk menampung pembacaan data mahasiswa }
Mhs : File of DataMhs

ALGORITMA:
Open(Mhs, 2)       { buka arsip Mhs untuk perekaman }
read(Msiswa.NIM) { baca NIM mahasiswa pertama, mungkin 9999 }
while(Msiswa.NIM ≠ 9999) do
      read(Msiswa.Nama, Msiswa.IP)
Fwrite(Mhs, Msiswa) { rekam Msiswa ke dalam arsip Mhs}
read(Msiswa.NIM)
endwhile
{ Msiswa.NIM = 9999 }
Close(Mhs
```

7.5 Pembacaan Arsip Beruntun

Proses pembacaan arsip beruntun merupakan kebalikan dari proses pembuatan arsip. Langkah pertama dalam membaca arsip adalah menyiapkan arsip untuk pembacaan (menggunakan perintah Open dengan Kode = 1). Setiap kali akan membaca suatu rekaman di dalam arsip, kita harus memastikan bahwa tanda akhir arsip belum dicapai.

Rekaman dibaca satu per satu, dimulai dari rekaman pertama hingga rekaman yang diinginkan sudah ditemukan atau seluruh rekaman selesai dibaca. Karena kita tidak tahu jumlah data di dalam arsip, maka konstruksi pengulangan yang cocok adalah dengan WHILE seperti yang ditunjukkan pada Algoritma 13.

```
PROGRAM PembacaanArsip
{ Skema pembacaan arsip beruntun tanpa penanganan kasus kosong }

DEKLARASI
type Rekaman : TipeRekaman
Arsip : File of Rekaman
Rek : Rekaman
```

```
ALGORITMA:
Inisialisasi
Open(Arsip, 1)     { buka arsip untuk dibaca}

while not EOF(Arsip) do
      Fread (Arsip, Rek)
      Proses Rek
endwhile
{ EOF(Arsip) }

Terminasi
Close(Arsip)
```

Sebagai contoh, diberikan sebuah arsip Bil yang berisi data bilangan bulat (proses pengisian dapat dilihat pada Contoh 1). Semua data di dalam arsip Bil dibaca kemudian ditampilkan ke layar, Algoritmanya adalah

```
PROGRAM BacaArsipBilanganBulat
{ Mencetak isi arsip ke layar peraga. Data dibaca dari arsip. }

DEKLARASI
Bil : File of integer
n : integer { banyaknya bilangan bulat}
x : integer        { peubah rekaman}

ALGORITMA:
Open(Bil, 1)       { buka arsip untuk pembacaan}
while not EOF(Bil) do
      Fread (Bil, x)     { baca rekaman pertama }
      write(x)
endwhile
{ EOF(Bil)

Close(Bil)   { tutup arsip }
```

7.6 Mengenali Alur Perulangan dalam Percabangan

Sebelumnya sudah dijelaskan terkait dengan alur percabangan dalam perulangan secara spesifik. Sekarang kita akan belajar hal yang sebaliknya, yaitu alur perulangan dalam percabangan. Pada struktur perulangan, perulangan sendiri merupakan struktur dimana terdapat proses pengulangan perintah yang sama sebanyak n kali. Struktur ini merupakan salah satu kelebihan yang dimiliki oleh mesin komputer. Sebagai contoh untuk mencetak teks “Algoritma Pemrograman” sebanyak 100 kali pada layar monitor, hanya diperlukan beberapa baris perintah meggunakan teknik atau struktur perulangan tersebut. Tanpa harus menuliskan perintah yang sama sebanyak 100 kali (Uce Indahyanti et al, 2020).

Untuk mengenali alur perulangan dalam perabangan kita perlu mempelajari penerapannya melalui rekursifitas atau proses rekursif beserta turunan-turunannya. Selain itu, untuk mempelajari lebih dalam kita juga perlu memahami dalam penerapan atau implementasinya dalam studi kasus riil. Pada bab ini kita akan belajar terkait dengan rekursifitas, yaitu salah satu tools yang sangat penting dalam pemrograman. Disebut sangat penting karena rekursif menyediakan teknik penyelesaian persoalan yang di dalamnya mengandung definisi persoalan itu sendiri. J.S. Rohl, seorang dosen di Universitas Western Australia, menyatakan bahwa rekursif adalah cinderella-nya teknik pemrograman (J.S. Rohl, 1984). Sayangnya, rekursif merupakan materi yang paling sulit dimengerti oleh pemula pemrograman. Pembahasan materi algoritma rekursif menggunakan banyak contoh agar pembaca mempunyai pemahaman yang kuat tentang rekursif

7.7 Proses Rekursif

Banyak objek di dalam matematika yang didefinisikan dengan cara menyatakan suatu proses (algoritma) yang menghasilkan objek tersebut. Sebagai contoh, π diperoleh dengan membagi keliling lingkaran (K) dengan diameternya (d). Dengan kata lain, proses (algoritma) untuk memperoleh nilai π dinyatakan sebagai berikut:

1. baca keliling lingkaran, K
2. baca diameter lingkaran, d
3. hitung π = K/ d

Yang jelas, proses tersebut harus berhenti dengan memberikan hasil yang didefinisikan.

Contoh objek lain yang diperoleh dari suatu proses adalah faktorial. Faktorial dari bilangan bulat tak-negatif n didefinisikan sebagai berikut:

$$n! = \begin{cases} 1, & ,n = 0 \\ 1\ x\ 2\ x\ \ldots, x(n-1)\ x\ n, & ,n > 0 \end{cases}$$

Sebagai contoh,

0! = 1

1! = 1

2! = 1 x 2

3! = 1 x 2 x 3

4! = 1 x 2 x 3 x 4

Proses untuk menghitung faktorial dari bilangan bulat n tak-negatif dinyatakan dalam algoritma berikut ini:

7.8 Definisi Rekursif

Definisi rekursif diturunkan secara matematik. Definisi yang tidak formal menyatakan bahwa sebuah objek dikatakan rekursif jika ia didefinisikan menjadi lebih sederhana dalam terminologi dirinya sendirinya.

Dalam kehidupan sehari-hari banyak terdapat objek yang rekursif. Tahukah Anda bahwa, kalau diamati dengan saksama, daun pakis (Gambar 7.3) dibentuk oleh ranting-ranting daun yang mempunyai pola yang mirip dengan daun pakis itu sendiri. Setiap ranting daun disusun lagi oleh ranting daun dengan pola yang mirip. Hal yang sama juga tampak pada pohon cemara. Objek rekursif yang khas ini disebut fraktal. Dalam bidang grafik dan seni, fraktal dimanfaatkan untuk membangkitkan gambar-gambar yang indah dan menawan.

Gambar 7. 3 Fraktal tanaman pakis

Daya guna rekursif terletak pada kemampuannya mendefinisikan sekumpulan objek yang tidak terbatas dengan sebuah pernyataan terbatas (Nicklaus Wirth, 1976). Sejumlah perhitungan yang tidak berhingga misalnya, dapat digambarkan dengan algoritma rekursif (teks algoritma adalah terbatas), seolah-olah algoritma tersebut mengandung pengulangan yang tidak "tampak" secara eksplislt.

Seperti yang sudah dinyatakan pada bagian awal bab ini, rekursif merupakan kakas yang sangat penting di dalam matematika dan pemrograman. Dalam matematika, banyak dltemukan fungsi yang rekursif, misalnya polinom Chebysev berikut:

$$n! = \begin{cases} 1, & ,n = 0 \\ x, & ,n = 1 \\ 2xT(n-1,x) - T(n-2,x), & ,lainnya \end{cases}$$

Berikut ini diberikan contoh-contoh fungsi rekursif yang lain:

1. Sembarang fungsi rekursif:
   (a) F(x) = 0 , x = 0
   (b) F(x) = 2F (x - 1) + x2 , x ≠ 0

2. Turunan fungsi:
   (a) $\frac{d(x^2)}{dx} = 2x, \frac{d(5x)}{dx} = 5$
   (b) $\frac{d(x^2+5x)}{dx} = \frac{d(x^2)}{dx} + \frac{d(5x)}{dx}$

3. Bilangan asli:
   (a) 1 adalah bilangan asli
   (b) Suksesor bilangan asli adalah bilangan asli

4. Polinom interpolasi Newton pn(x) yang melalui titik-titik (x0, y0), (x1, y1), ..., (xn, yn) didefinisikan sebagai berikut:
   (a) P0(x) = y0
   (b) Pn(x) = pn-1(x) + an(x – x0) (x – x1)...(x – xn-1)

5. Dalam pemrograman, salah satu struktur data yang penting adalah pohon biner (binary tree). Sebuah pohon biner didefinisikan sebagai berikut:
   (a) Kosong adalah pohon (disebut juga pohon kosong)
   (b) Jika t1 dan t2 pohon, maka 

adalah pohon

Jika diperhatikan dari contoh-contoh di atas adalah maka definisi rekursif disusun oleh dua bagian:

(i) Basis
Bagian yang berisi kasus yang terdefinisi secara eksplisit. Bagian ini juga sekaligus menghentikan rekursif (dan memberikan sebuah nilai yang terdefinisi pada fungsi rekursif).

(ii) Rekurens
Bagian ini mendefinisikan objek dalam terminologi dirinya sendiri

Bagian (b) menyatakan bahwa definisi rekursif memungkinkan komputasi yang tidak berhenti. Pada setiap kali pendefinisian, akan dihasilkan "bentuk" yang makin sederhana, sehingga pada suatu saat pendefinisian itu akan berhenti. Bagian (a) berisi kasus yang menghentikan pendefinisian rekursif.

Tinjau kembali perhitungan n! secara rekursif. Dengan mengingat kembali definisi rekursif dari faktorlal:

(i) n! = 1 , jika n = 0 {basis}

(ii) n! = n x (n -1)! , jika n > 0 {rekurens}

maka 5! dihitung dengan langkah berikut:

(1) 5! = 5 x 4! (rekurens)

(2) 4! = 4 x 3!

(3) 3! = 3 x 2!

(4) 2! = 2 x 1!

(5) 1! = 1 x 0!

(6) 0! = 1

Pada baris (6) kita memperoleh nilai yang terdefinisi secara langsung dan bukan faktorial dari bilangan lainnya. Dengan melakukan runut-balik (backtrack) dari baris (6) ke baris (1), kita mendapatkan nilai pada setiap baris untuk menghitung hasil pada baris sebelumnya:

(6') 0! = 1

(5') 1! = 1 x 0! = 1 x 1 = 1

(4') 2! = 2 x 1! = 2 x 1 = 2

(3') 3! = 3 x 2! = 3 x 2 = 6

(2') 4! = 4 x 3! = 4 x 6 = 24

(1') 5! = 5 x 4! = 5 x 24 = 120

Jadi, 5! = 120.

Algoritma rekursif untuk menghitung n! sebagai berikut:

```
function Fak(input n:integer)→integer
{ mengembalikan nilai n!. Algoritma rekursif.
      basis       : jika n = 0, maka 0! = 1
      rekurens    : jika n > 0, maka n! = n x (n-1)|
}

DEKLARASI
-
ALGORITMA:
if n = 0 then
      return 1                      {basis}
else
      return n* Faktorial (n - 1)   {rekurens}
endif
```

Perhatikanlah bahwa dalam bagian rekurens terkandung pengulangan yang implisit. Pengulangan itu ditunjukkan oleh pemanggilan kembali fungsi Fak dengan parameter yang nilainya terus berkurang. (n -1). Bagaimana pemanggilan rekursif dilakukan adalah tanggung jawab compiler. Pemrogram hanya perlu memikirkan bagaimana skema algoritma rekursifnya. Fungsi Fak di atas memberikan kita contoh skema algoritma rekursif yang paling sederhana, yaitu rekursif lanjar (linier recursion), karena hanya ada satu pemanggilan rekursif.

7.9 Implementasi Kombinasi Perulangan dalam Percabangan (Contoh Imlementasi Proses Rekursif)

Untuk memahami implementasi kombinasi perulangan dalam percabangan kita perlu mempelajari penerapannya melalui contoh-contoh implementasi proses rekursif serta memahami cara kerja compiler menangani rekursif. Dengan memahami cara compiler menangani program rekursif dapat membantu kita untuk:

1. Meminimumkan jumlah parameter yang dilewatkan pada waktu pemanggilan,
2. Mengubah algoritma rekursif menjadi program iteratif dengan cara meniru (simulasi) mekanisme pelaksanaan program rekursif.

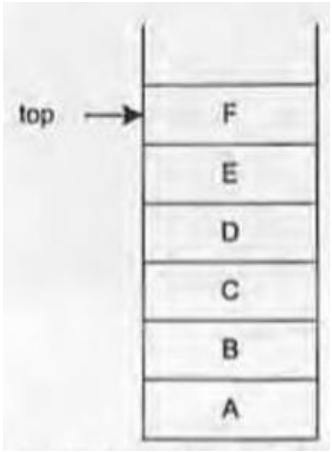
Pertama-tama kita harus mengetahui mekanisme pemanggilan prosedur/fungsi yang ditangani oleh compiler. Ketika sebuah prosedur/fungsi dipanggil, area data di memori dialokasikan untuk pemanggilan tersebut. Area data ini akan menyimpan:

1. Parameter yang dilewatkan waktu pemanggilan.
2. Semua peubah lokal di dalam prosedur/fungsi.
3. Alamat kembali (return address), yaitu alamat instruksi berikutnya di dalam program pemanggil setelah prosedur/fungsi selesai dilaksanakan.

Setelah mengisi data area, kendali program dipindahkan ke prosedur/fungsi. Semua acuan terhadap nilai parameter dan peubah lokal adalah nilai yang terdapat di dalam data area. Bila prosedur/fungsi selesai dilaksanakan, alamat kembalinya

diambil dari area data, kendali program bercabang ke alamat tersebut, dan alokasi memori untuk area data dibebaskan.

Tiap compiler berbeda-beda dalam mengimplemantasikan area data, tetapi pada umumnya area data diimplementasikan dalam bentuk tumpukan (stack). Tumpukan (stack) adalah sekumpulan elemen di mana semua penambahan (insert) elemen atau penghapusan (delete) elemen hanya dapat dilakukan pada sebuah sisi yang disebut top. Gambar 7.4 adalah contoh sebuah tumpukan yang berisi 6 buah elemen. Elemen yang paling atas disebut ditunjuk oleh puncak (top). Penambahan elemen baru hanya dapat dilakukan di atas top, dan elemen yang dihapus hanyalah elemen pada posisi top.



Gambar 7. 4 Sebuah tumpukan dengan 6 buah elemen

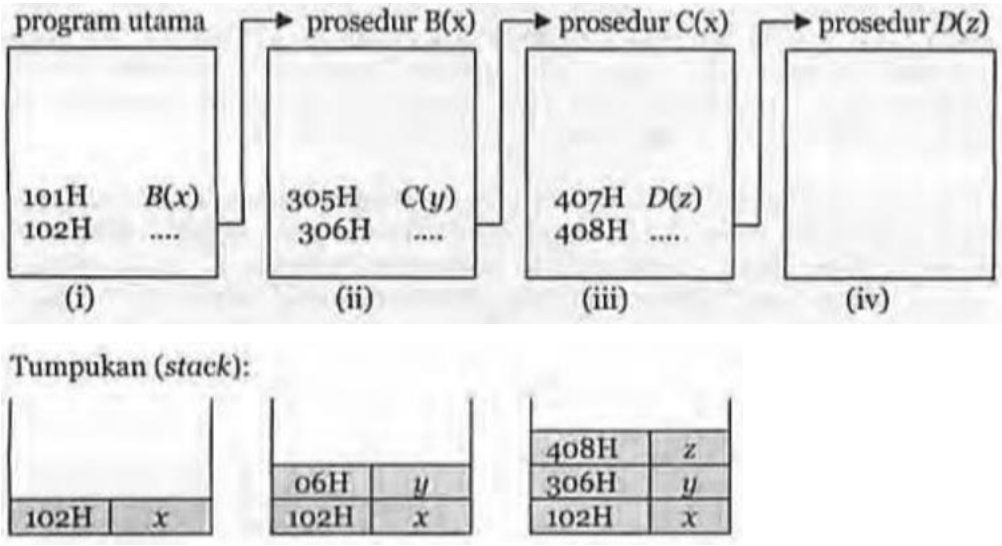
Struktur dan tumpukan mempunyai dua buah operasi primitif. yaitu push dan pop.

Push(s, x) : menambahkan (insert) elemen x sebagai elemen puncak (top) dari tumpukan s

Pop(s) : mengambil (delele) elemen puncak (top) dari tumpukan s

Dengan menggunakan tumpukan, suatu prosedur/fungsi tidak kehilangan kendali ketika memanggil prosedur/fungsi lainnya. Pada setiap kali pemanggilan, akan dialokasikan memori (area data) untuk elernen baru tumpukan, kemudian elemen ini di-push ke dalam tumpukan. Ketika prosedur/fungsi selesai dilaksanakan, elemen di puncak tumpukan di-pop, alamat kembalinya diambil, dan

alokasi memori untuk elemen tersebut dibebaskan (Garnbar 4 dan 5). Angka heksadesimal 101H, 102H, 305H, dan seterusnya menyatakan alamat setiap instruksi di dalam memori,



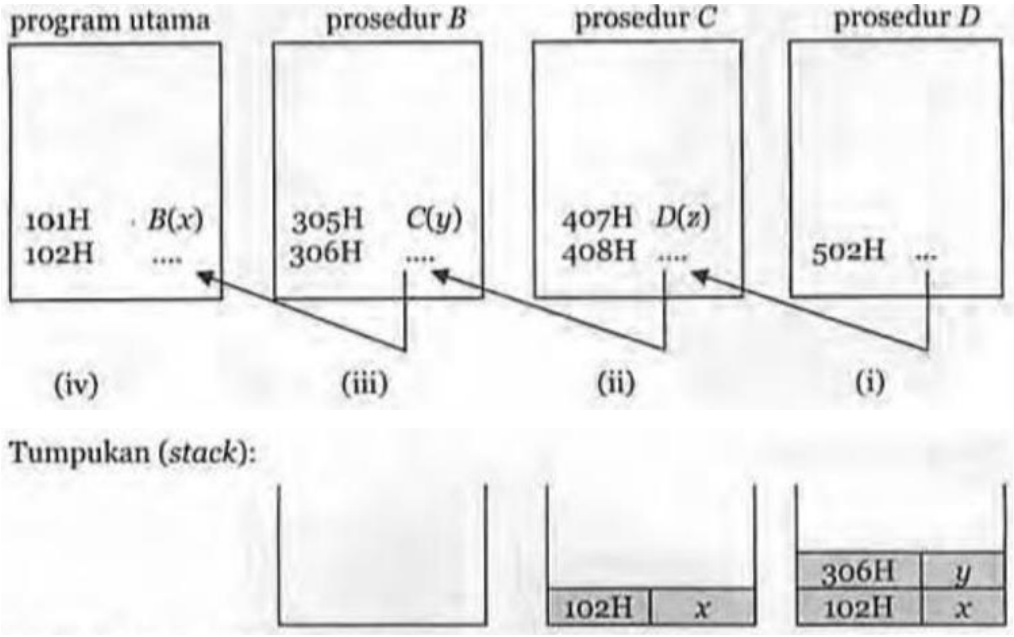
Gambar 7. 5 Skema pemanggilan prosedur

Pada gambar 7.5 dijelaskan bahwa angka 101H. 102H, dan seterusnya, menyatakan alamat instruksi di dalam memori (dalam heksadesimal). Anak panah menyatakan kendali program. Setiap kali pemanggilan, elemen baru di-push ke dalam tumpukan.

Perhatikan bahwa tumpukan ini tidak "terlihat" dari sisi pengguna program, sehingga disebut juga tumpukan implisit. Pengalokasian tumpukan beserta pengoperasianya ditangani oleh compiler, bukan lagi urusan pemrogram.

Mekanisme yang sama juga berlaku pada waktu pemanggilan prosedur/fungsi rekursif. Setiap kali prosedur rekursif dipanggil, alamat kembali, semua peubah lokal, dan semua parameter lokal dimasukkan (push) ke dalarn tumpukan. Acuan terhadap parameter lokal atau peubah lokal adalah nilai yang terdapat pada elemen di puncak tumpukan.

Apabila prosedur rekursif yang "terdalam" selesai dilaksanakan, elemen di puncak tumpukan diambil (pop), alokasi memorinya dibebaskan, dan elemen puncak sekarang dipakai sebagai acuan parameter lokal dan peubah lokal untuk pelaksanaan prosedur rekursif sebelah luarnya,



Gambar 7. 6 Setelah prosedur selesai dilaksanakan

Gambar 7.6 menjelaskan bahwa setelah prosedur selesai dilaksanakan, elemen di puncak tumpukan di-pop, kendali program kemudian berpindah ke instruksi yang ditunjukkan oleh alamat kembali. Alokasi memori elemen yang di-pop dibebaskan.

Pada Contoh 4, digambarkan jika kita ingin mencetak setiap digit bilangan bulat n. Misalnya jika n = 375, maka dicetak 3, 7, 5 {keluaran di layar: 375}

TulisBiasa(n):

(i) jika n < 10, cetak (n) {basis}

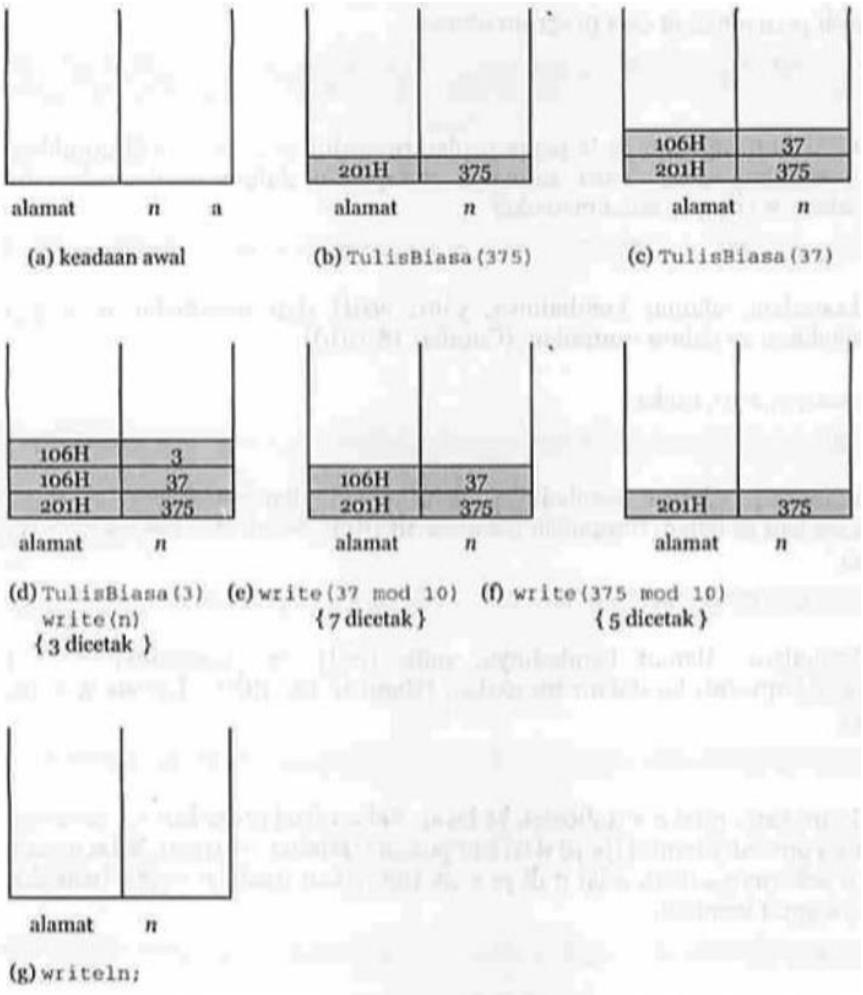
(ii) jika n ≥ 10, {rekurens}

- `TulisBiasa(n div 10)`
- `cetak(n mod 10)`

Algoritma rekursif untuk menghitung n! sebagai berikut:

```
098H  procedure TulisBiasa(input n:integer)
      { mencetak digit bilangan bulat dari depan ke belakang
            K. Awal : n terdefinisi
            K. Akhir : digit-digit n dicetak ke layar
                  basis       : jika n < 10, maka cetak(n)
                  rekurens : jika n > 10, maka
                        -     TulisBiasa (n div 10)
                        -     cetak(n div 10)
            }
```

```
      DEKLARASI
099H  -
100H
101H  ALGORITMA:
102H        if n < 10 then
103H              write(n)
104H        else
105H              TulisBiasa(n div 10);
106H              write(n mod 10);
107H        endif
```



Gambar 7. 7 Isi tumpukan pada pemanggilan TulisBiasa

Contoh pemanggilan dari program utama:

```
200H  TulisBiasa(375)
201H  writeln;
```

Ilustrasi isi tumpukan pada pemanggilan prosedur TulisBiasa ditunjukkan pada Gambar 7.7. Pada mulanya, tumpukan dalam keadaan kosong (Gambar 7(a)). Ketika instruksi

```
TulisBiasa(375)
```

dilaksanakan, alamat kembalinya, yaitu 201H dan parameter n = 375 dimasukkan ke dalam tumpukan (Gambar 7(b)).

Karena 375 > 10, maka

```
TulisBiasa(375 div 10)
```

dilaksanakan. Alamat kembalinya, yaitu 106H dan parameter n = 37 dimasukkan ke dalam tumpukan (Gambar 7(c)). Sekali lagi, karena 37 > 10, maka

```
TulisBiasa(37 div 10)
```

dilaksanakan. Alamat kembalinya, yaitu 106H dan parameter n = 3 dimasukkan ke dalam tumpukan (Gambar 7(d)). Karena 3 < 10, maka

```
write(n)
```

dilaksanakan; nilai n = 3 dicetak ke layar. Lalu keluar dari prosedur TulisBiasa, elemen puncak diambil (pop) dari tumpukan (Gambar 7(e)). Nilai n yang diacu sekarang adalah nilai n di puncak tumpukan (yaitu n = 37). Instruksi pada alamat kembali,

```
write(n mod 10)
```

dilaksanakan, dan digit 7 dicetak ke layar. Keluar dari prosedur TulisBiasa, elemen puncak diambil (pop) dari tumpukan (Gambar 7(f). Nilai n yang diacu sekarang adalah nilai n di puncak tumpukan (yaitu n = 375). Instruksi pada alamat kembali,

dilaksanakan, dan digit 5 dicetak ke layar. Keluar dari prosedur TulisBiasa, elemen puncak diambil (pop) dari tumpukan (Gambar 7(g). Instruksi pada alamat kembali (yaitu 201H),

```
writeln
```

dllaksanakan. Program selesai dilaksanakan

POST TEST

Soal dan Pembahasan tentang Materi Kombinasi Struktur Algoritma Pemrograman

1. Apa yang kamu ketahui tentang arsip beruntun?
2. Bagaimana contoh implementasi dari arsip beruntun?
3. Apa yang kamu ketahui tentang rekursif?
4. Bagaimana contoh implementasi dari rekursif?