

Seri Buku Persiapan Seleksi Tim Olimpiade Komputer Indonesia

Buku Untuk Siswa dan Guru

Konsep Dasar

Pemrograman Prosedural

(dilengkapi contoh soal dan pembahasan)

Disusun Oleh :
Tim Pembina TOKI

Judul Buku	: Konsep Dasar Pemrograman Prosedural (Dilengkapi contoh soal dan pembahasan)
Penyusun	: Tim Pembina TOKI
Kontributor	: Yohanes Nugroho, SKom
Penyunting	: TOKI
Disain Cover	: TOKI
Diterbitkan oleh	: Bagian Proyek Pengembangan Wawasan Keilmuan Direktorat Pendidikan Menengah Umum, Direktorat Jenderal Pendidikan Dasar dan Menengah, Departemen Pendidikan Nasional RI
Cetakan Pertama	: 2004

Seri Buku Periapan Seleksi Olimpiade Komputer Indonesia

Buku 1 Untuk Siswa	: Referensi Pemrograman Bahasa Pascal (<i>menggunakan free pascal ver. 1.0.10</i>)
Buku 2 Untuk Siswa dan Guru	: Konsep Dasar Pemrograman Prosedural (<i>dilengkapi contoh soal dan pembahasan</i>)
Buku 3 Untuk Guru	: Aspek Pedagogi Pengajaran Pemrograman Pertama (<i>Menggunakan Bahasa Pascal</i>)

Copyleft :

- ☞ *Seluruh isi dalam buku ini diijinkan untuk diperbanyak dan disebarluaskan sejauh untuk kepentingan pendidikan dan pengajaran, dengan tetap mencantumkan sumbernya.*
- ☞ *Buku ini juga dapat didownload dari situs web TOKI di www2.toki.or.id dan situs-situs pendukung lainnya*

Kata Pengantar

Sejak keikutsertaan Indonesia dalam ajang International Olympiad in Informatics pada tahun 1997, prestasi siswa-siswa dalam ajang tersebut cukup membanggakan. Hingga tahun 2004 ini (keikutsertaan yang ke 9) secara total Tim Olimpiade Komputer Indonesia telah mengumpulkan 1 Medali Emas, 6 Medali Perak dan 7 Medali Perunggu.

Sejauh ini Tim Olimpiade Komputer Indonesia yang dikirim ke ajang IOI masih didominasi oleh siswa-siswa di wilayah Jawa, terutama DKI Jakarta, Jawa Barat dan Jawa Timur, hal ini menunjukkan bahwa masih terdapat ketimpangan pengetahuan dan pembinaan di bidang Komputer/Informatika terhadap siswa-siswa di daerah lain. Hal ini dapat terjadi salah satunya adalah karena keterbatasan ketersediaan materi pelajaran computer, khususnya yang mengarah kepada materi-materi yang dilombakan dalam ajang Olimpiade Komputer Indonesia.

Buku ini diterbitkan dalam beberapa seri untuk siswa maupun untuk guru dimaksudkan agar dapat menjadi bahan pelajaran bagi siswa dan panduan pengajaran bagi guru yang memadai untuk mempersiapkan siswa menghadapi rangkaian seleksi Olimpiade Komputer Indonesia. Dengan harapan bahwa kesempatan dan peluang bagi siswa-siswa di seluruh Indonesia menjadi lebih terbuka.

Terima Kasih kepada semua pihak yang telah memberikan kontribusinya sehingga penerbitan buku ini dapat terwujud. Besar harapan kami buku ini dapat bermanfaat bagi semua pihak. Kritik dan saran yang membangun sangat kami harapkan.

Jakarta, Oktober 2004

Daftar Isi

Kata Pengantar	iii
Daftar Isi	iv
Pendahuluan	v
Bab 1. PARADIGMAN PEMROGRAMAN DAN MEKANISME EKSEKUSI PROGRAM	1
Bab 2. PEMROGRAMAN PROSEDURAL	7
Bab 3. PENGENALAN PROGRAM UTAMA PASCAL	9
Bab 4. VARIABEL DAN TIPE	11
Bab 5. ANALISA KASUS	23
Bab 6. PENGULANGAN	27
Bab 7. SUBPROGRAM	31
Bab 8. TIPE BENTUKAN	37
Bab 9. FILE	43
Bab 10. REKURENS	49
LAMPIRAN 1. CONTOH SOAL DAN PEMBAHASAN, UJIAN TEORI PEMROGRAMAN PASCAL	51
LAMPIRAN 2. CONTOH SOAL DAN PEMBAHASAN, UJIAN PRAKTEK PEMROGRAMAN PASCAL	73
INDEX	99

Pendahuluan

Mengenai Buku ini

Buku ini ditulis untuk mengajarkan konsep dasar pemrograman. Buku ini tidak mengajarkan penggunaan bahasa tertentu meskipun contoh yang ditulis menggunakan bahasa Pascal. Buku ini dirancang untuk dipakai dengan Buku Kecil Bahasa Pascal sebagai pendamping. Contoh program dalam buku ini relatif sedikit, dan sebagian besar contohnya bisa dilihat di buku pendamping tersebut.

Buku ini dirancang sebagai buku pegangan untuk murid yang akan belajar pemrograman untuk mengikuti seleksi TOKI, dan merupakan bagian dari set buku TOKI yang lain. Buku ini mengacu banyak pada Kisi-Kisi TOKI dan Buku Panduan FreePascal yang juga diterbitkan oleh TOKI.

Apa yang dicakup oleh buku ini

Buku ini akan membahas konsep pemrograman prosedural yang umum, yang ada pada semua bahasa pemrograman yang ditujukan untuk pemrograman dengan paradigma prosedural (mengenai paradigma sendiri, Anda bisa melihat bagian lain dari buku ini). Konsep prosedural yang dibahas meliputi: tipe, variabel, kondisional, loop, fungsi, dan prosedur.

Apa yang tidak dicakup dalam buku ini

Buku ini tidak membahas mengenai pemrograman Pascal, dan juga tidak membahas mengenai algoritma rumit dan problem solving. Andaikan buku ini adalah buku pelajaran untuk bahasa manusia, maka buku ini tidak mengajarkan bahasa Inggris atau Perancis, tidak juga mengajarkan sastra untuk bahasa Indonesia, buku ini hanya mengajarkan konsep bahasa (misalnya konsep kata benda, nomina, dan lain-lain).

Problem solving merupakan masalah lain dalam pemrograman yang juga tidak dibahas oleh buku ini. Buku ini hanya berisi konstruksi dasar pemrograman yang dapat digunakan untuk menyelesaikan masalah sederhana, namun tidak untuk masalah kompleks. Dianalogikan dengan bahasa manusia, buku ini tidak mengajarkan membuat karya sastra seperti puisi dan novel meskipun akan membuat Anda bisa menyusun kalimat dari kata-kata sederhana.

Bab 1

PARADIGMA PEMROGRAMAN DAN MEKANISME EKSEKUSI PROGRAM

Bab ini akan memperkenalkan paradigma pemrograman dan mekanisme eksekusi program. Di sini akan dijelaskan secara singkat apa itu paradigma pemrograman, beberapa contoh paradigma yang ada, serta sekilas mengenai paradigma prosedural. Di bagian berikutnya dibahas mengenai bagaimana suatu program dieksekusi di komputer, serta peran interpreter, kompilator, dan debugger dalam pembuatan program.

Paradigma Pemrograman

Paradigma pemrograman adalah bagaimana cara pandang kita terhadap penyelesaian masalah pemrograman (atau sudut “serang” kita dalam menyelesaikan suatu masalah pemrograman). Ada banyak cara untuk menyelesaikan suatu masalah, sehingga ada banyak paradigma yang ada.

Beberapa contoh paradigma pemrograman yang ada saat ini adalah: prosedural, fungsional, deklaratif, dan objek. Secara singkat dapat dikatakan bahwa:

- ? paradigma prosedural memandang penyelesaian masalah sebagai hasil dari serangkaian langkah yang menyelesaikan sub masalah
- ? paradigma fungsional memandang penyelesaian masalah sebagai komposisi fungsi yang memetakan masalah ke jawaban
- ? paradigma deklaratif memandang penyelesaian masalah adalah pekerjaan komputer yang dilakukan melalui inferensi terhadap fakta
- ? paradigma objek memandang penyelesaian masalah sebagai hasil interaksi dari objek (objek dalam konsep ini merupakan representasi objek di dunia nyata)

Buku ini tidak akan membahas secara detail masing-masing paradigma, paradigma yang telah disebutkan sekilas di atas hanya untuk memberikan gambaran bahwa solusi untuk suatu masalah tidak harus dilakukan secara prosedural.

Paradigma prosedural merupakan paradigma yang sangat intuitif sehingga mudah dipelajari. Dalam paradigma prosedural masalah diselesaikan dengan menggunakan langkah-langkah yang berurutan yang disebut sebagai suatu algoritma. Selain sangat intuitif bagi programmer, cara penyelesaian prosedural ini juga merupakan cara yang paling alami bagi komputer (secara hardware, komputer bekerja secara sekuensial atau berurutan).

Mekanisme Eksekusi Program

Komputer merupakan benda yang “bodoh” yang hanya bisa menjalankan instruksi dalam bahasa mesin, bukan bahasa manusia. Komputer yang pertama diprogram langsung dengan menuliskan bahasa mesin ke dalam komputer, seiring berlalunya waktu, hal itu dirasakan tidak efisien sehingga diciptakanlah bahasa assembly, berupa kata-kata singkat yang lebih mudah diingat dibanding dengan kode yang harus dimasukkan langsung. Bahasa assembly sebenarnya tidak jauh dari bahasa mesin namun sudah cukup untuk membantu programmer menulis program dengan lebih mudah. Bahasa assembly ini disebut sebagai bahasa tingkat rendah.

Pada tahun enam puluhan, para ahli mulai banyak membuat bahasa yang lebih mudah dimengerti oleh manusia, bahasa tersebut disebut sebagai bahasa tingkat tinggi. Ada banyak bahasa yang diciptakan, bahkan sangat banyak, namun sedikit yang bertahan hingga saat ini. Tapi semua bahasa tersebut memiliki kesamaan yaitu bahwa mereka tidak bisa langsung dimengerti oleh komputer sehingga perlu diterjemahkan ke dalam bahasa mesin.

Penerjemahan dapat dilakukan dengan menggunakan program (yang pada awalnya dulu ditulis dengan bahasa assembly) yang bisa berupa sebuah interpreter atau sebuah kompilator (atau gabungan dari keduanya). Program penerjemah tersebut akan memeriksa sintaks (format penulisan) apakah benar atau tidak, lalu menerjemahkan program tersebut ke dalam bahasa mesin.

Interpreter

Interpreter adalah suatu program komputer yang mampu menerjemahkan program dari bahasa tingkat tinggi yang dimengerti oleh manusia dan langsung menjalankan program tersebut. Kerja

interpreter seperti penerjemah untuk turis yang langsung menerjemahkan kalimat demi kalimat yang dikatakan oleh sang turis.

Setiap kali kita membutuhkan program tersebut, maka interpreter akan bekerja menerjemahkan program dari bahasa tingkat tinggi ke bahasa mesin untuk dieksekusi. Jadi siklus kerja ketika kita membuat program dengan interpreter adalah: tulis/edit program, eksekusi.

Kompilator

Kompilator adalah suatu program komputer yang membaca seluruh program dari bahasa tingkat tinggi yang dimengerti oleh manusia dan kemudian menerjemahkan keseluruhan program tersebut dalam bahasa mesin. Program yang sudah diterjemahkan tersebut akhirnya akan dijalankan oleh komputer. Kerja kompilator seperti penerjemah buku yang akan menerjemahkan seluruh buku sekaligus, sehingga setiap orang bisa mengerti makna buku dalam bentuk terjemahannya.

Kompilator hanya perlu bekerja sekali saja menerjemahkan bahasa tingkat tinggi ke bahasa mesin, dan jika kita membutuhkan kembali program tersebut, kita hanya perlu menjalankannya, kompilator tidak perlu bekerja lagi. Jadi siklus kerja jika kita memakai kompilator adalah: tulis/edit program, kompilasi, eksekusi

Kompilator vs Interpreter

Apakah suatu bahasa diinterpretasi atau dikompilasi bergantung pada ketersediaan interpreter atau kompilator untuk bahasa tersebut. Sebagai contoh, kita tidak dapat mengatakan bahwa bahasa BASIC adalah bahasa yang diinterpretasi, karena ada juga kompilator untuk bahasa BASIC.

Interpreter dan kompilator masing-masing memiliki keuntungan dan kerugian. Kelebihan interpreter adalah Pengembangan program lebih cepat, tidak perlu melakukan kompilasi yang mungkin butuh waktu lama, namun kerugiannya setiap kali program perlu dijalankan, interpreter harus bekerja lagi, sehingga kecepatan eksekusi program menjadi kurang jika dibanding dengan kompilator.

Sebaliknya penggunaan kompilator memungkinkan kita membentuk program yang dapat langsung dijalankan dengan cepat (karena sudah dalam bahasa mesin), namun dibutuhkan waktu yang relatif lama dalam pengembangan programnya.

Kompilator + Interpreter

Meskipun tidak terlalu penting dalam pembahasan buku ini, namun perlu diketahui bahwa ada bahasa yang dikompilasi namun tidak ke dalam bahasa mesin (ke bahasa antara), lalu diinterpretasi oleh suatu interpreter untuk menjalankannya. Sebagian pekerjaan interpreter (memvalidasi program) sudah dilakukan oleh kompilator, sehingga interpreter hanya perlu mengeksekusi program saja. Contoh bahasa yang menggunakan pendekatan ini adalah: Java, C#, dan VB.NET

Debugger

Kesalahan pertama yang ditemukan pada salah satu komputer pertama (yang saat itu masih sangat besar) adalah karena adanya serangga/kutu (*bug*) yang menyebabkan komputer tidak bekerja. Sejak saat itu semua kesalahan, baik di bidang hardware maupun software komputer disebut dengan bug (istilah ini lebih umum di bidang software dibanding hardware).

Proses untuk menemukan kesalahan program disebut juga dengan proses pencarian bug (istilah proses ini adalah debug). Dalam pencarian kesalahan ini terkadang diperlukan program pembantu yang dinamakan debugger. Program ini akan membantu programmer untuk melihat bagaimana eksekusi program dilakukan oleh komputer, dan melihat kesalahan yang mungkin ada ketika program sedang berjalan.

Editor, Kompilator, dan IDE

Untuk memasukkan program ke dalam komputer, kita perlu tools yang dinamakan editor. Editor adalah program yang mampu menerima teks dari manusia, dan menyimpannya ke dalam bentuk digital yang dimengerti komputer. Editor juga memungkinkan kita melakukan koreksi terhadap pengetikan yang kita lakukan (menghapus teks, menyalin teks, dan lain-lain).

Untuk menjalankan program yang sudah kita ketikkan, kita akan membutuhkan kompilator atau interpreter. Pada bahasa Pascal, kompilator lebih umum dipakai. Perlu diperhatikan bahwa editor dan kompilator adalah dua program yang terpisah dan berbeda.

Sebuah IDE (Integrated Development Environment) adalah program yang menggabungkan fungsi editor dan kompilator (serta terkadang debugger) dalam satu paket. IDE saat ini semakin populer, bahkan banyak orang yang menyangka bahwa IDE sama dengan Kompilator. Sebuah IDE mungkin

saja sekaligus memiliki fungsi kompilator, tapi tidak selalu demikian, terkadang IDE hanya menyediakan fungsi editor, dan akan memanggil kompilator yang sesungguhnya ketika kita akan mengkompilasi program.

Aneka Implementasi Kompilator Pascal

Sama seperti mobil, yang memiliki fungsi yang sama namun memiliki banyak merk, saat ini ada banyak merk kompilator. Setiap kompilator ini memiliki fungsi yang sama namun memiliki perbedaan dalam hal detailnya. Perbedaan kompilator ini ada karena beberapa hal:

- ? Perbedaan sistem operasi menyebabkan sifat kompilator berbeda
- ? Kurang detailnya standar bahasa Pascal menyebabkan interpretasi yang berbeda terhadap standar Pascal
- ? Masing-masing pembuat kompilator menambahkan fitur di luar standar Pascal untuk mempermudah pembuat program

Secara umum, semua fitur standar bahasa Pascal akan didukung oleh aneka kompilator yang ada saat ini. Jika Anda terkadang menemukan ada program yang tidak berjalan di suatu sistem operasi atau di suatu kompilator tertentu, maka Anda perlu memeriksa apakah program yang Anda buat memenuhi standar Pascal.

FreePascal

Dari aneka kompilator yang ada, FreePascal dipilih sebagai acuan dalam buku ini. FreePascal dipilih dengan beberapa alasan:

- FreePascal tersedia gratis, dan bersifat open source, sehingga kode sumber kompilatornya sendiri bisa dilihat dan dipelajari
- Dokumentasi FreePascal juga tersedia gratis
- FreePascal merupakan kompilator resmi yang dipakai pada IOI (International Olympiad in Informatics/Olimpiade Informatika Internasional)
- FreePascal memenuhi standar Pascal

Meskipun kompilator FreePascal banyak diacu dalam buku ini, buku ini bisa dipakai bersama kompilator yang lain (bahkan dapat dipakai untuk bahasa pemrograman selain Pascal), namun perlu diperhatikan bahwa setiap kompilator memiliki perbedaan, dan hal tersebut harus dikonsultasikan pada manual masing-masing kompilator.

Bab 2

PEMROGRAMAN PROSEDURAL

Pemrograman dalam paradigma prosedural dilakukan dengan memberikan serangkaian perintah yang berurutan. Dalam bab ini akan dibahas hal-hal yang menjadi dasar dalam pemrograman prosedural, meliputi definisi algoritma dan konstruktor pemrograman prosedural, serta konsep Input, Proses, dan Output yang sangat lazim dalam dunia pemrograman prosedural.

Algoritma

Algoritma adalah serangkaian langkah-langkah yang tepat, terperinci, dan terbatas untuk menyelesaikan suatu masalah. Langkah yang tepat artinya serangkaian langkah tersebut selalu benar untuk menyelesaikan masalah yang diberikan. Langkah yang tidak memberikan hasil yang benar untuk domain masalah yang diberikan bukanlah sebuah algoritma.

Langkah yang terperinci artinya setiap langkah diberikan secara detail dan dapat dieksekusi oleh komputer, instruksi seperti “angkat sedikit ke kiri” merupakan contoh instruksi yang tidak tepat, karena “sedikit” tidak menyatakan sesuatu yang tepat.

Langkah yang diberikan harus terbatas, artinya suatu saat langkah harus berhenti, jika langkah tidak pernah berhenti (misalnya: “ambil air, masukkan ke bak mandi, ulangi ambil air, dan seterusnya”) maka serangkaian langkah itu tidak disebut sebagai algoritma (jika: “ambil air, masukkan ke bak mandi, ulangi ambil air sampai bak mandi penuh”, maka bisa disebut algoritma, namun langkah ambil air, masukkan ke bak mandi, harus diperinci).

Konstruktor (elemen) Pemrograman Prosedural

Elemen bahasa pemrograman prosedural yang penting adalah:

1. Program utama
2. Tipe
3. Konstanta

4. Variabel
5. Ekspresi, operator, dan operand
6. Struktur Data
7. Instruksi dasar
8. Program Moduler
9. File eksternal
10. Rekurens

Konstruktor ini tidak untuk dipelajari secara berurutan, namun semua perlu dipelajari dan dimengerti untuk dapat membuat program dengan baik.

Input, Proses, dan Output

Sekumpulan aksi dalam pemrograman prosedural bisa dibagi menjadi tiga bagian penting yaitu: input, proses, dan output. Bagian input, proses, dan output dikerjakan secara sekuensial, dan dalam setiap bagian mungkin akan ada input, proses, dan output.

Bab 3

PENGENALAN PROGRAM UTAMA DALAM PASCAL

Mengenal program utama dalam suatu bahasa akan memberikan gambaran global bagaimana menulis program dalam bahasa tersebut. Program dalam bahasa Pascal memiliki format dasar seperti berikut (format lebih lengkap akan diberikan secara bertahap):

```
Program namaprogram;  
begin  
    (*bagian program utama*)  
end.
```

Kata-kata yang ditebalkan merupakan kata kunci (keyword) dalam bahasa Pascal. Kata kunci adalah kata-kata baku dalam bahasa Pascal yang memiliki arti khusus, kata-kata tersebut harus kita pakai sesuai dengan makna yang sudah diberikan oleh Pascal, dan tidak bisa kita ubah.

Bagian pertama berisi nama program. Bagian ini tidak wajib ada di kompilator Pascal yang baru, namun sebaiknya tetap ditulis, kita bisa memberi nama program dengan kata kunci program, misalnya: program bilanganprima;

Bagian berikutnya adalah blok program utama yang ditandai dengan begin dan end. Perhatikan bahwa setelah end ada tanda titik yang menyatakan akhir program. Bagian di dalam tanda kurung diikuti oleh bintang/asterisk, (*seperti ini*) merupakan komentar program yang tidak akan diproses oleh kompilator (hanya untuk dibaca oleh manusia, sebagai tambahan keterangan).

Program yang diberikan di atas tidak melakukan apa-apa, meskipun dapat dikompilasi dan dijalankan. Program sederhana yang dapat kita buat berikutnya adalah program “hello world” yang akan mencetak kalimat “hello world” ke layar komputer.

```
Program hello;  
begin  
    writeln(' Hello World' );  
end.
```

Program tersebut juga ada pada buku Contoh Program kecil dalam bahasa Pascal. Beberapa hal yang perlu diperhatikan adalah:

- ? Pascal tidak membedakan case atau kapitalisasi huruf (jadi writeln dengan WRITELN dianggap sama)
- ? Setelah setiap instruksi harus ada titik koma (titik koma adalah pemisah antar instruksi), kecuali instruksi terakhir sebelum end, boleh ada titik koma, boleh juga tidak
- ? Writeln adalah salah satu prosedur standar Pascal

Batasan Penamaan Identifier

Identifier adalah nama yang diberikan untuk fungsi, prosedur, tipe, variabel, dan untuk program. Semua nama yang disebutkan memiliki batasan tergantung pada kompilator yang digunakan, namun umumnya:

- ? Nama tidak boleh diawali dengan angka, 2you adalah identifier yang tidak valid
- ? Nama boleh berupa huruf yang digabung dengan angka, tapi tidak boleh diawali angka: its4you adalah nama yang valid
- ? Nama yang hanya terdiri dari huruf saja pasti valid (sampai panjang tertentu, tergantung kompilator)
- ? Nama biasanya boleh mengandung tanda underscore (garis bawah seperti ini: nama_orang)

Perhatian: pilihlah nama yang singkat namun deskriptif untuk menamai apapun dalam program.

Bab 4

VARIABEL DAN TIPE

Konsep variabel dalam pemrograman mirip dengan konsep variabel dalam matematika. Variabel adalah suatu nama yang dapat diasosiasikan dengan sebuah nilai yang dapat kita manipulasi. Seperti dalam matematika, kita mengenal tipe untuk suatu variabel, misalnya $1 = x < 5 \mid x \in \mathbb{R}$ yang artinya x adalah suatu variabel bertipe real (domain x adalah bilangan real), dengan range (jangkauan) 1 sampai 5.

Contoh penggunaan variabel yang sederhana ada pada contoh program kecil BACA.PAS. Perhatikan bahwa deklarasi variabel (pernyataan variabel apa memiliki tipe apa ada pada bagian sebelum blok utama begin end, seperti ini:

```
Program namaprogram;
var
    nama_variabel : tipevariabel;
    nama_variabel2 : tipevariabel2;
begin
    (*bagian program utama*)
end.
```

Sintaks yang lebih lengkap dapat dilihat pada buku pemrograman bahasa Pascal. Perhatikan bahwa ketika dideklarasikan sebuah variabel belum terdefinisi nilainya (sudah memiliki nilai, tapi tidak bisa diprediksi nilai apa yang ada). Salah satu cara untuk memberi nilai variabel adalah melalui assignment.

Assignment

Assignment adalah pemberian nilai kepada variabel. Assignment memberikan nilai pada ruas kiri sesuai dengan hasil nilai di ruas kanan. Misalnya jika a adalah sebuah variabel yang tipenya bilangan bulat:

```
var a: integer;  
begin  
    a := 2;  
end.
```

akan memberikan nilai 2 pada variabel a. Untuk melihat nilai a, kita bisa meng*output*kan nilai tersebut dengan instruksi `writeln`, seperti ini:

```
writeln(a);
```

setelah instruksi `a:=2`. Perhatikan bahwa tipe di sebelah kanan harus sama dengan tipe di sebelah kiri (pembahasan mengenai assignment untuk tipe yang berbeda dapat dilihat di bagian Kompatibilitas Tipe)

Representasi Tipe

Komputer hanya bisa memproses angka, sehingga semua tipe data dalam komputer akan diproses dalam bentuk bilangan integer. Bahkan kata-kata yang muncul dalam komputer juga diproses sebagai bilangan. Untuk masing-masing tipe yang dibahas di sini, akan diberikan juga representasi di dalam komputer untuk masing-masing tipe untuk mengetahui batasan dari setiap tipe.

Pemahaman representasi tipe ini penting untuk mengetahui mengapa suatu tipe terbatas, mengapa hasil suatu operasi seperti yang dijelaskan. Sebenarnya representasi tipe ini menjadi bahasan dalam kuliah atau pelajaran arsitektur komputer, namun karena tidak ada buku lain yang digunakan yang memuat hal tersebut, pembahasan representasi akan digabung dalam penjelasan tipe.

Perlu ditekankan bahwa pengetahuan mengenai representasi tipe tidak terlalu penting dalam pemrograman, yang penting adalah hanya mengetahui batasan dari setiap tipe yang ada, sehingga dapat memilih tipe yang tepat ketika membuat program.

Konstanta

Konstanta adalah suatu nilai yang tidak berubah. Contohnya pi (p), adalah konstanta yang digunakan sebagai perbandingan keliling lingkaran terhadap diameternya, dan e adalah konstanta bilangan *euler*. Dalam Pascal dan dalam semua bahasa prosedural lain sebuah nama boleh diberi nilai yang tidak akan diubah di dalam program, nama ini disebut sebagai konstanta. Kata kunci yang dipakai dalam Pascal adalah `const`.

Const

```
PI = 3.14;
```

Penggunaan konstanta yang lebih lengkap dapat dilihat dalam buku panduan FreePascal (buku 1).

Variabel bertipe Dasar

Variabel bertipe dasar adalah variabel yang memiliki tipe yang sudah didefinisikan oleh suatu bahasa. Tipe dasar yang sudah didefinisikan Pascal meliputi: Integer, String, Karakter, Boolean, dan Real. Variabel bertipe dasar akan sangat banyak digunakan, dan merupakan elemen pembentuk tipe bentukan, sehingga penguasaan tipe dasar ini sangat penting.

Input dan Output Variabel bertipe Dasar

Setiap bahasa pemrograman umumnya sudah menyediakan cara untuk melakukan input dan output tipe dasar. Dalam Pascal, output tipe dasar dilakukan dengan prosedur `write` dan `writeln`. Beda kedua prosedur tersebut adalah: `write` tidak memajukan kursor ke baris berikutnya sedangkan `writeln` memajukan kursor ke baris berikutnya (ln di sini berarti line atau baris).

Untuk menuliskan atau mengoutputkan variabel bertipe dasar, gunakan `write` atau `writeln` seperti ini:

```
write(var1);
```

atau agar lebih jelas, gunakan string untuk menjelaskan arti output:

```
writeln('Nilai var1 adalah ', var1);
```

Sebaliknya untuk membaca input dari pengguna, gunakan `read` atau `readln`. Contoh penggunaan `read` adalah:

```
read(a);
```

dimana `a` adalah suatu variabel dengan tipe dasar manapun. Untuk memperjelas, sebaiknya sebelum `read` perlu diberikan informasi kepada pengguna mengenai apa yang harus dilakukan:

```
write("Masukkan sebuah angka:");  
read(a);
```

read dan readln memiliki fungsi yang sama untuk variabel bukan string, pada variabel string, read hanya membaca 1 kata sedangkan readln bisa membaca 1 kalimat.

Integer

Integer adalah suatu tipe bilangan bulat (negatif, positif, dan nol). Integer dipakai dalam kebanyakan operasi matematika dan loop, bahkan beberapa prosesor tidak memiliki kemampuan perhitungan bilangan real sehingga semua perhitungan numerik dilakukan dengan integer.

Representasi integer dalam komputer

Integer memiliki representasi yang sederhana dalam komputer. Komputer memandang integer sebagai nilai dari serangkaian bilangan biner. Namun komputer tidak memproses satu bit demi satu bit, tapi per blok bit yang umumnya terdiri dari 8 bit (dikenal sebagai 1 byte atau binary eight). Ada beberapa tipe integer, tipe integer paling sederhana adalah byte yaitu integer 8 bit yang unsigned (tidak bisa menyimpan nilai negatif, atau hanya bisa menampung tipe bilangan asli/natural), dan “pasangannya” yaitu tipe shortint yang tipe integer 8 bit yang bertanda (signed atau bilangan bulat).

Bilangan integer 8 bit artinya diperlukan memori sebesar 8 bit untuk menyimpan tipe tersebut, nilai yang bisa disimpan adalah 0000 0000 sampai dengan 1111 1111. Jika dipandang sebagai tipe unsigned, maka nilai desimal untuk 8 bit tersebut adalah 0 sampai dengan 255. Untuk bisa menyimpan nilai negatif dalam biner biasanya digunakan representasi yang disebut sebagai komplement 2.

Komplement 2 adalah kebalikan dari representasi desimal ditambah dengan 1. Untuk menyimpan -5 desimal misalnya, kita cari representasi untuk 5 yaitu 0000 0101, kita balik (not-kan) menjadi 1111 1010, dan kita tambahkan 1 menjadi 11111011, sehingga representasi -5 dalam biner adalah 11111011. Untuk menyimpan bilangan positif 5, tetap digunakan 0000 0101. Dengan cara komplement 2 maka nilai yang bisa ditampung dalam 8 bit adalah dari -128 sampai dengan 127

Tipe integer lain adalah **integer** yang berupa tipe integer 32 bit (atau 16 bit pada Pascal di DOS), dan long yang berupa tipe integer 64 bit. Perbedaan dari masing-masing tipe tersebut adalah kemampuannya dalam jangkauan angka yang bisa disimpan.

Operasi Terhadap Integer

Ada beberapa operasi dasar untuk integer yaitu: kali, bagi, tambah, kurang, dan mod. Operasi kali (disimbolkan dengan bintang/asterisk *), tambah (+), dan kurang (-) sudah jelas dan berlaku seperti yang sudah dipelajari. $5 + 5 = 10$, $2 * 3 = 6$, dan $3 - 2 = 1$. Operasi pembagian (div) akan memberikan hasil pembagian yang dibulatkan, jadi $4 \text{ div } 2 = 2$, dan $5 \text{ div } 2 = 2$. Operasi mod memberikan sisa dari hasil bagi sehingga $5 \text{ mod } 2 = 1$, dan $2 \text{ mod } 5 = 2$ ($2/5$ jika dibulatkan adalah nol, sisanya adalah $2 - 0 * 5 = 2$). Dalam beberapa bahasa pemrograman operasi mod mungkin tidak ada.

Ada satu operator unary (operator yang memakai satu operand) untuk integer yaitu minus dan plus. Contoh ekspresinya: $-5 * 2 = -10$. Sebelum angka 5 ada operator minus yang menyatakan bilangan negatif.

Operasi perbandingan integer meliputi, > (lebih besar), < (kurang dari), <= (kurang dari atau sama dengan), >= (lebih dari atau sama dengan), tidak sama dengan (<>) dan = (sama dengan). Operasi perbandingan memiliki semantik yang sama seperti yang dipelajari dalam matematika (artinya: sama seperti di matematika $6 < 5$).

Urutan Operator (Precedence of operator) Integer

Ekspresi $2 * 4 + 5 * 2$ akan menghasilkan 18 karena operasi perkalian dilakukan lebih dahulu jika dibandingkan dengan penjumlahan. Dalam hal ini dikatakan bahwa precedence kali lebih tinggi dari bagi. Umumnya precedence of operator untuk integer di aneka bahasa adalah sama.

Karakter

Di dalam Pascal dan aneka bahasa lain, dikenal tipe data char yang bisa menampung satu karakter. Satu karakter adalah satu huruf, atau satu angka, atau simbol. Sebuah variabel bertipe karakter hanya boleh diisi dengan satu simbol saja, seperti ini:

```
var c: char;  
begin  
    c := 'A'; (* c berisi huruf A *)  
end.
```

Representasi karakter dalam komputer

Dalam bahasa Pascal standar. Sebuah karakter adalah sebuah bilangan integer 8 bit yang memiliki nilai dari 0 sampai 255. Setiap nilai ini dipetakan dalam suatu simbol, misalnya nilai 65 berkaitan dengan simbol A, 66 dengan simbol B, dan seterusnya. Pemetaan ini tergantung pada encoding yang dipakai (dan encoding ini bergantung pada wilayah atau bahasa tertentu), karakter 0-127 memiliki pemetaan standar yang disebut dan biasanya disebut dengan karakter ASCII (American Standard Code For Information Interchange), sedangkan sisanya bergantung pada encoding yang dipakai (misalnya orang Perancis memetakan menjadi beraneka huruf yang memiliki accent, orang Rusia memetakan dalam alfabet mereka).

Dalam Pascal, ada fungsi `ord` (ordinal) yang bisa memetakan dari karakter menjadi nilainya, dan fungsi `chr` (character) yang melakukan proses kebalikannya. Contoh pemakaian `ord`:

```
writeln(ord('A'));
```

akan mencetak angka 65, dan contoh pemakaian `chr`

```
writeln(chr(65));
```

akan mencetak huruf A.

UNICODE

Dalam sistem operasi yang modern, terdapat jenis karakter yang disebut dengan UNICODE, yang lebih kompleks dari karakter sederhana. Karakter UNICODE dapat merepresentasikan hampir semua karakter yang masih dipakai di bumi (dan yang sudah tidak dipakai, seperti hieroglyph). Pada dasarnya penyimpanan UNICODE tetap memakai angka tapi memakai jumlah bit yang lebih dari karakter biasa (bervariasi dari 8 – 64 bit, tergantung pada encoding yang dipakai)

Operasi Terhadap char

Sebuah tipe karakter tidak bisa dioperasikan, tidak bisa dijumlahkan ataupun dikurangkan. Tipe char hanya bisa dioperasikan jika dikonversi menjadi integer dengan `ord`. Sebuah tipe char bisa digabungkan pada sebuah tipe string.

Sebuah karakter bisa dibandingkan dengan karakter lain, hasil perbandingan adalah perbandingan nilai ordinal karakter tersebut. Operasi perbandingan yang bisa dilakukan sama dengan yang bisa dilakukan dengan integer. Contoh: 'A' < 'B', karena ordinal 'A' adalah kurang dari 'B', namun 'b' > 'A', karena ordinal 'b' (b kecil) lebih dari ordinal 'A' (A kapital).

Perhatian

Manipulasi karakter dengan melakukan konversi ke integer merupakan hal yang sangat tidak disarankan. Kebiasaan melakukan hal ini juga berbahaya jika suatu saat Anda beralih ke UNICODE, karena hasil manipulasi serupa mungkin tidak bisa diprediksi.

String

String sebenarnya adalah tipe variabel dasar yang cukup kompleks, namun akan sering dipakai, dalam bagian ini representasi string tidak akan dibahas, hanya penggunaannya saja. Suatu literal integer dapat ditulis seperti biasa: 5 atau 6, sedangkan literal string harus ditulis dengan diapit tanda kutip tunggal, seperti ini:

```
'ini string'
```

Jika ada tanda kutip di dalam string, digunakan dua kutip tunggal, jadi untuk menuliskan string "don't do that", harus seperti ini:

```
'don' 't do that'
```

Operasi Terhadap String

Ada beberapa operasi untuk string, namun yang akan dibahas di bagian ini hanyalah instruksi penambahan atau konkatenasi string. String dapat digabungkan seperti ini:

```
program concat_string;
var s1, s2, s3 : string
begin
  s1 := 'hello';
  s2 := 'world';
  s3 := s1 + ' ' + s2;
  writeln(s3);
end.
```

Nilai string s3 adalah 'hello world' yang merupakan gabungan dari string s1, spasi, dan string s2. Sebuah karakter bisa juga digabung atau disambungkan dengan sebuah karakter:

```
var
    s: string;
    c: char;
begin
    writeln('Masukkan sebuah huruf:'); readln(c);
    s:= 'Huruf yang Anda masukkan adalah '+c;
    writeln(s);
end.
```

Operasi terhadap string yang lain dilakukan dengan menggunakan fungsi-fungsi string yang tergantung pada kompilator Pascal. Fungsi-fungsi string yang tersedia untuk FreePascal dapat dilihat pada buku FreePascal.

Boolean

Boolean adalah suatu tipe data yang hanya memiliki nilai true (benar) dan false (salah). Tipe boolean sangat diperlukan dalam kondisi perulangan dan kondisional (menggunakan if). Ekspresi yang menghasilkan boolean bisa berupa ekspresi dengan tipe-tipe yang terdiri dari tipe boolean, bisa juga berupa ekspresi dari tipe lain.

Contoh ekspresi yang menghasilkan boolean adalah $6 > 5$, karena bilangan 6 memang lebih besar dari 5 maka nilai ekspresi tersebut adalah true, sedangkan $6 < 2$ adalah ekspresi yang nilainya false. Operator untuk boolean ada beberapa yaitu: and, or, not, dan xor

- OR** akan menghasilkan true jika salah satu operannya bernilai true
- AND** akan menghasilkan true jika kedua operannya bernilai true
- XOR** akan menghasilkan true jika operannya memiliki nilai boolean yang berbeda
- NOT** akan menghasilkan nilai boolean kebalikan dari nilai yang diberikan

Dalam Pascal, dan berbagai bahasa umumnya nilai true lebih besar dari nilai false (true > false adalah true)

Representasi Boolean dalam komputer

Nilai boolean disimpan sebagai angka 0 untuk false dan bukan nol untuk true. Dalam bahasa Pascal, boolean tidak bisa dioperasikan sebagai integer, namun dalam bahasa lain (C misalnya), boolean dan integer adalah dua nilai yang bisa dipertukarkan.

Urutan Operator (Precedence of operator) Boolean

Sama seperti pada integer, ada urutan operator pada operasi boolean. Urutan untuk operator boolean dapat dilihat pada buku panduan Free Pascal.

Operasi Boolean pada Bilangan Integer

Bilangan integer dapat dioperasikan dengan operator boolean, proses operasi dilakukan dengan melihat representasi bit pada bilangan integer, dan melakukan operasi boolean yang bersesuaian terhadap bit tersebut (bit 1 dianggap true, dan 0 dianggap false).

Contoh: $1 \text{ or } 2 = 3$

representasi biner untuk 1 desimal adalah (kita ambil 8 bit) 00000001, sedangkan representasi biner untuk 2 desimal adalah 00000010, maka jika kita or -kan masing-masing bit, hasilnya adalah 00000011, yang artinya 3 desimal

Real

Real adalah tipe yang dapat menampung bilangan real. Tipe ini bisa menampung bilangan dengan suatu nilai di belakang koma dengan presisi tertentu (lihat bagian representasi real).

Representasi Real dalam komputer

Ada banyak representasi real dalam komputer, namun umumnya real direpresentasikan dalam bentuk:

$$A * 2^{+/- b}$$

Misalnya nilai 0.5 akan direpresentasikan sebagai $1 * 2^{-1}$ ($1/2 = 0.5$). Perhatikan bahwa tidak semua nilai bisa direpresentasikan dengan tepat dengan cara ini, nilai $1/3$ misalnya tidak bisa dinyatakan dengan tepat dalam bentuk perkalian tersebut. Baik nilai A maupun nilai B memiliki presisi integer yang terbatas.

Operasi Terhadap Real

Operasi yang bisa dilakukan terhadap real meliputi: tambah, kali, minus (sama seperti integer), dan pembagian (memakai simbol / yang menghasilkan bilangan real), operasi MOD tidak terdefinisi untuk real.

Operasi perbandingan selain = (sama dengan) dan tidak sama dengan (<>) bisa dilakukan terhadap real (>, <, >=, <=, <>). Operasi sama dengan dan tidak sama dengan sebenarnya bisa dilakukan (tidak dibatasi oleh bahasa Pascal), namun dalam kasus tertentu hasilnya mungkin tidak sesuai dengan yang diharapkan.

Cara yang benar untuk membandingkan 2 buah real adalah dengan menetapkan nilai epsilon yang kecil yang dipakai untuk membandingkan:

```
if (a - b < epsilon) then
begin
    writeln("A sama dengan B");
end else
begin
    writeln("A tidak sama dengan B");
end;
```

(mengenai penggunaan kalimat kondisional, lihat bab Analisa Kasus)

Kompatibilitas Tipe

Sebuah variabel bertipe string tentu tidak bisa dijumlahkan dengan variabel bertipe integer, karena tipe string tidak kompatibel dengan tipe integer. Kedua tipe itu sangat berbeda, sehingga masalah kompatibilitas tipe itu sudah jelas. Namun dalam kasus tertentu, kompatibilitas tipe mungkin tidak terlalu jelas dan harus diperhatikan dengan baik.

Contohnya, variabel bertipe real tidak bisa langsung diassign pada variabel bertipe integer, karena variabel bertipe real mungkin mengandung pecahan, tapi hal yang sebaliknya bisa dilakukan. Variabel bertipe real dengan presisi yang tinggi (misalnya double di Pascal) tidak bisa diassign pada variabel bertipe real dengan presisi yang lebih rendah (tipe real biasa di Pascal), karena presisinya bisa hilang.

Permasalahan menjadi lebih kompleks pada ekspresi yang melibatkan banyak variabel dengan berbagai jenis tipe. Contohnya dalam ekspresi ini:

```
a : integer;
b : integer;
d : integer;
c : real;
begin
    a :=1;
    b :=2;
    c :=a/b;
    d :=a/b;
end.
```

Ekspresi a/b akan menghasilkan tipe real, sehingga instruksi terakhir adalah salah dan tidak bisa dikompilasi (tipe di ruas kanan tidak sama dengan tipe di ruas kiri).

Konversi Antar Tipe Sederhana

Suatu tipe dapat dikonversi ke tipe lain meskipun biasanya konversi ini tidak sempurna. Contoh konversi tipe adalah: konversi dari tipe dengan presisi yang lebih tinggi ke yang lebih rendah (integer ke byte misalnya, konversi dari tipe bertanda ke tidak bertanda, konversi dari string ke integer, konversi dari real ke integer).

Sebagian proses konversi dapat dilakukan melalui proses yang disebut dengan *casting*. Casting merupakan proses konversi sederhana yang hanya melakukan perubahan representasi internal tanpa kehilangan makna. Contohnya casting bisa dilakukan untuk mengubah sebuah shortint menjadi byte, atau byte menjadi integer, namun sebuah bilangan real tidak bisa dicast menjadi sebuah integer.

Cara melakukan casting adalah dengan memberikan ekspresi yang akan dicasting di dalam nama tipe tujuan casting:

```
var
    a: byte;
    b: real;
begin
    a := 3 div 2;
    writeln(integer(a));
    b := real(a);
    writeln(real(b));
    writeln(real(1/2));
end.
```

Konversi a ke real bisa dilakukan karena bilangan integer pasti bisa dikonversi ke real. Konversi integer(b) akan menghasilkan error karena dalam konversi real ke integer ada beberapa hal yang bisa

dilakukan, apakah akan dibulatkan ke atas, atau dibulatkan ke bawah, atau semua angka di belakang koma akan dihilangkan.

Konversi dari tipe yang lebih presisi lebih tinggi, atau yang memiliki range yang lebih besar ke tipe yang presisi atau range yang lebih kecil akan memiliki hasil yang tidak terdefinisi dalam kasus tertentu. Misalnya i adalah sebuah integer, dan b adalah sebuah byte, konversi dari i ke b (integer ke byte) akan menghasilkan hasil yang benar jika i memiliki nilai antara 0 sampai 255, jika i memiliki nilai di luar itu maka nilai b tidak bisa diprediksi.

Konversi tipe dengan fungsi

Konversi tipe yang melibatkan tipe yang berbeda jauh dapat dipandang sebagai konversi tipe yang tidak sederhana. Konversi ini melibatkan penggunaan fungsi khusus yang menjadi bagian dari bahasa, atau fungsi tambahan pendukung bahasa tersebut.

Contoh fungsi yang adalah `ceil` untuk membulatkan bilangan ke atas (6.7 menjadi 7), `floor` untuk membulatkan bilangan real ke bawah (6.7 menjadi 6), dan `round` untuk membulatkan bilangan real ke nilai yang terdekat (6.7 menjadi 7).

Fungsi atau prosedur konversi juga tersedia untuk mengubah integer/real menjadi string dan sebaliknya. Dalam kasus konversi string ke numerik, ada kemungkinan bahwa string tidak mengandung karakter angka, tapi karakter lain sehingga konversi gagal.

Fungsi konversi yang disebutkan bukan merupakan standar Pascal, sehingga perlu dikonsultasikan pada dokumentasi kompilator yang dipakai. Untuk FreePascal, informasi ini juga bisa diperiksa di buku FreePascal.

Bab 5

ANALISA KASUS

Instruksi (kalimat) kondisional adalah instruksi yang dilakukan berdasarkan kondisi boolean tertentu. Instruksi ini adalah instruksi yang sangat penting, yang memungkinkan komputer untuk bisa “berpikir” dengan mengambil aksi berdasarkan kondisi boolean tertentu.

IF Sederhana

Bentuk paling sederhana dari pernyataan kondisional dalam setiap bahasa adalah if - then. Contoh:

```
if (a>b) then writeln("A lebih besar dari B");
```

bagian setelah if disebut dengan bagian kondisi dan bagian setelah then adalah bagian aksi. Jika aksi lebih dari satu instruksi maka bisa dikelompokkan dalam blok begin end. Seperti ini:

```
if (a>b) then  
begin  
    writeln("Kesimpulan: ");  
    writeln("A lebih besar dari B");  
end;
```

IF dengan ELSE

Selain if - then, ada bentuk kondisional lain yaitu if - then - else. Kita bisa mengubah instruksi semacam ini:

```
if (a>b) then writeln("A lebih besar dari B");  
if (a<=b) then writeln("A kurang dari atau sama dengan B");
```

menjadi:

```
if (a>b) then writeln("A lebih besar dari B") else ("A kurang dari atau sama dengan B");
```

Bagian else akan dieksekusi jika bagian kondisi tidak dipenuhi, atau dengan kata lain, bagian else dieksekusi jika kondisi yang terjadi adalah komplemen dari kondisi di bagian if.

Perhatikan: untuk memperjelas program, sebaiknya semua instruksi if yang memiliki else ditulis dengan komentar setelah else (untuk menjelaskan kondisi apa yang ditangani oleh else kepada pembaca program) seperti ini:

```
if (a>b) then
begin
    writeln("A lebih besar dari B");
end else (* a<= b *)
begin
    ("A kurang dari atau sama dengan B");
end;
```

IF untuk banyak kondisi

Instruksi if boleh digabung dengan banyak instruksi if menangani kondisi yang kompleks (banyak kondisi dan banyak aksi), seperti ini:

```
if (kondisi_1) then
begin
    aksi 1;
end
else if (kondisi_2) then
begin
    aksi 2;
end else (* kondisi 1 dan 2 tidak dipenuhi *)
begin
    aksi_n;
end.
```

Case untuk banyak aksi

Selain IF, ada bentuk analisa kasus yang digunakan untuk menangani banyak kondisi yang ada dalam bentuk (a adalah suatu ekspresi yang menghasilkan tipe enumerasi, dan k1 .. kn adalah konstanta):

```
if (a = k1 ) then
begin
    aksi 1;
end else if (a = k2) then
begin
    aksi 2;
end else if (a=k3) then
begin
    aksi 3;
end else
begin
    Aksi_n;
end;
```

Bentuk ini adalah bentuk case of, seperti ini:

```
case a of
k1 :
  begin
    aksi 1;
  end;
k2 :
  begin
    aksi 2;
  end;
k3:
  begin
    aksi 3;
  end;
else
  begin
    aksi_n;
  end;
end;
```

Hal yang perlu diperhatikan dalam analisa kasus dengan case adalah bahwa ekspresi yang bisa ditangani dalam case hanya yang nilainya bisa dienumerasi, yaitu integer (dan tipe range), boolean, dan tipe enum (dijelaskan pada bagian struktur data)

Bab 6

PENGULANGAN

Pengulangan adalah suatu cara untuk mengulangi eksekusi blok instruksi sampai suatu kondisi tertentu dipenuhi.

Bentuk Pengulangan FOR

Bentuk pengulangan ini adalah yang paling sederhana, jika kita ingin melakukan aksi sebanyak N kali, maka kita bisa (dan sebaiknya) menggunakan bentuk pengulangan ini. Pengulangan dilakukan mulai dari suatu indeks awal, sampai dengan indeks akhir, misalnya:

```
for i:=1 to 5 do writeln('Hello');
```

akan mencetak 5 kali kata hello. Indeks loop (dalam kasus ini i) berguna karena dapat diakses di dalam loop, seperti ini:

```
for i:=1 to 5 do writeln(i);
```

yang akan mencetak angka 1 sampai dengan 5.

Bentuk loop ini sangat sederhana, karena kita bisa memastikan bahwa loop pasti akan berhenti, yaitu ketika nilai indeks sudah sama dengan nilai akhirnya.

Perhatian: Anda disarankan untuk TIDAK mengubah INDEKS LOOP dalam program, seperti ini:

```
for i:=1 to 5 do i:=i+2;
```

Karena hasilnya mungkin tidak dapat diprediksi

Bentuk Pengulangan WHILE - DO

Terkadang jumlah pengulangan yang akan dilakukan belum diketahui di awal sehingga kita tidak bisa menggunakan bentuk perulangan for. Bentuk loop ini adalah

```
while (kondisi) do  
begin  
    aksi ;  
end;
```

Aksi dalam while akan dilakukan selama kondisi memiliki nilai true, dan aksi mungkin tidak dilakukan sama sekali jika di awal loop kondisi sudah bernilai false. Hal yang perlu diperhatikan dalam loop ini adalah kepastian bahwa loop akan berhenti (bahwa suatu saat kondisi akan bernilai false).

Bentuk Pengulangan REPEAT - UNTIL

Bentuk pengulangan while do mungkin tidak pernah mengeksekusi instruksi di dalam loop satu kali pun jika kondisi sudah tidak terpenuhi. Jika diinginkan agar minimal sebuah aksi dieksekusi sekali, maka gunakan bentuk pengulangan repeat until

```
repeat  
    aksi ;  
until kondisi ;
```

Aksi akan diulangi sampai kondisi bernilai true.

bentuk ini sama saja dengan:

```
aksi ;  
while (not kondisi) do  
begin  
    aksi ;  
end;
```

Karena loop repeat bisa dikonversi ke bentuk while, maka hal yang perlu diperhatikan sama yaitu bahwa loop pasti akan berhenti.

NESTED LOOP (Loop Bersarang)

Nested loop adalah loop yang memiliki loop di dalam bagian aksinya. Nested loop dapat dibuat dengan aneka bentuk loop yang sudah ada. Jadi kita bisa membuat loop for di dalam loop for, atau loop while dalam repeat until. Jumlah loop di dalam loop bisa sangat banyak (hanya dibatasi oleh kemampuan kompilator dalam menangani loop bersarang).

Nested loop for dapat memakai indeks loop di luarnya sebagai batas. Potongan program ini (dengan i dan j adalah integer):

```
for i:=1 to 5 do
begin
  for j:=1 to i do
  begin
    writeln("*");
  end;
end;
```

Akan mencetak:

```
*
**
***
****
*****
```

Invarian Loop

Invarian Loop adalah kondisi dalam loop yang selalu benar sepanjang loop tersebut berjalan. Contohnya dalam loop ini:

```
var i, jumlah : integer
begin
  i:=0;
  jumlah:=0;
  while (i<5) do
  begin
    jumlah:=jumlah + i;
    i:=i+1;
  end;
end.
```

Dalam loop ini invarian loop adalah jumlah akan selalu ditambahkan dengan nilai i saat ini, dan nilai i akan selalu bertambah dengan satu dari nilai pada loop berikutnya, sehingga loop dijamin akan berhenti.

Invarian loop untuk loop sederhana mudah diperiksa, namun di dalam loop mungkin ada analisa kasus yang akan mempersulit pemeriksaan apakah loop sudah benar.

Bab 7

SUBPROGRAM

Di saat program kita sudah menjadi besar kita akan mengalami kesulitan dalam mengatur kode program, jika semua kode tersebut disatukan. Untuk mengatasi hal tersebut, kita bisa menggunakan subprogram untuk membuat program kita terbagi menjadi beberapa bagian yang masing-masing lebih kecil dan lebih mudah dikelola.

Pemecahan program menjadi subprogram yang lebih kecil juga akan mempermudah kita jika ingin membuat program yang serupa, kita hanya perlu menyalin, atau memakai subprogram yang sudah ada di dalam program kita yang baru.

Dalam setiap bahasa pemrograman prosedural umumnya dikenal prosedur dan fungsi, dua buah bentuk subprogram yang umum. Fungsi adalah subprogram yang memetakan suatu nilai dari suatu domain ke domain lain, sedangkan prosedur adalah suatu subprogram yang melakukan aksi tertentu.

Prosedur

Prosedur memungkinkan kita membagi program dengan mengelompokkan aksi menjadi bagian-bagian yang jelas terpisah, prosedur juga memungkinkan kita memisahkan bagian program yang sering diulang sehingga tidak perlu ditulis ulang di setiap waktu.

Contoh penggunaan prosedur sederhana:

```
program testprocedure;
procedure CetakHeader;
begin
    writeln('TIM TOKI');
    writeln('-----');
end;
```

```

var
    c: char;
begin
    writeln('Cetak laporan harian atau bulanan: (H/B):');
    if (c='H') then
        begin
            CetakHeader;
            (*diteruskan dgn instruksi untuk mencetak laporan harian *)
        end
    else if (c='B') then
        begin
            CetakHeader;
            (*diteruskan dgn instruksi untuk mencetak laporan bulanan*)
        end
    else
        begin
            writeln('Pilihan salah');
        end;
end.

```

Perhatikan bahwa pemanggilan prosedur dilakukan dengan menggunakan nama prosedur, seperti ini:

```
CetakHeader;
```

Prosedur dengan Parameter

Agar prosedur lebih generik, maka prosedur bisa menerima parameter. Parameter prosedur berguna untuk menjadi input prosedur:

```

procedure cetakbintang(jumlahbintang: integer)
var i: integer
begin
    for i:=1 to jumlahbintang do write('* ');
end;

```

Prosedur di atas bisa dipanggil dengan:

```
cetakbintang(5);
```

yang akan mencetak 5 bintang (*****) di layar.

Scope Variabel

Di dalam setiap prosedur boleh ada deklarasi variabel, setiap prosedur boleh memiliki parameter. Variabel-variabel di dalam prosedur ini boleh memiliki nama yang sama. Nama variabel dalam sebuah prosedur hanya berlaku untuk prosedur tersebut, atau secara formal: scope variabel pada sebuah prosedur hanya pada prosedur tersebut.

Beberapa aturan:

- ? Nama parameter dan variabel dalam prosedur (disebut variabel lokal) tidak boleh sama
- ? Jika ada nama variabel global yang sama dengan variabel lokal atau nama variabel parameter, maka yang akan diakses adalah variabel lokal atau parameter
- ? Jika tidak ada nama variabel lokal yang sama dengan global, maka variabel global yang akan diakses

Lihat contoh mengenai program kecil SUBPRG.PAS dan LINGKUP.PAS

Prosedur dengan Parameter by Reference

Perhatikan bahwa dalam prosedur seperti ini:

```
procedure tambahsatu(angka: integer)
var i: integer
begin
    angka = angka+1;
end;
```

dan prosedur tersebut dipanggil dengan cara:

```
var
    jumlah: integer;
begin
    jumlah = 5;
    tambahsatu(jumlah);
    writeln(jumlah);
end.
```

Maka nilai jumlah tidak berubah, tetap 5. Hal ini dikarenakan passing parameter dilakukan by value, artinya nilai jumlah akan disalin, baru akan dipakai oleh prosedur, sehingga nilai jumlah sendiri tidak berubah (hanya salinannya yang berubah). Jika kita mengubah prosedur tersebut menjadi:

```
procedure tambah satu(var angka: integer)
var i: integer
begin
    angka := angka + 1;
end;
```

Perhatikan bahwa ada kata kunci `var` sebelum `angka`, hal itu menyatakan bahwa parameter `angka` dilewatkan berdasarkan referensi (berdasarkan variabel itu sendiri), sehingga sekarang program main yang sama akan menghasilkan angka 6. Perhatikan bahwa prosedur `tambah satu` tidak bisa dipanggil seperti ini:

```
tambah satu(5);
```

karena nilai 5 tidak bisa diubah oleh prosedur `tambah satu`. Dengan kata lain, parameter `var` hanyalah sesuatu yang boleh berada di ruas kiri assignment.

Fungsi

Sama seperti definisi dalam matematika, fungsi dalam pemrograman juga berarti pemetaan suatu nilai dari suatu domain ke domain yang lain. Pascal sudah menyediakan aneka fungsi standar, seperti fungsi trigonometri, fungsi kuadrat, dan lain-lain.

Dalam implementasinya fungsi hampir sama persis dengan prosedur (boleh memiliki variabel, parameter, dan lain-lain) dengan satu perbedaan yaitu fungsi dapat dan harus mengembalikan nilai.

Contoh sebuah fungsi adalah:

```
function kuadrat(x: integer): integer;
begin
    kuadrat := x * x;
end;
```

Perhatikan bahwa pada instruksi:

```
kuadrat := x * x;
```

Ruas kiri adalah nama fungsi itu sendiri, di sini nilai kembalian dari fungsi diberikan. Untuk memanggil fungsi ini:

```
var hasil : integer;  
begin  
    hasil := kuadrat(5);  
end.
```

Variabel hasil akan memiliki nilai 25

Bab 8

TIPE BENTUKAN

Tipe dasar sudah cukup untuk dapat dipakai memproses sebagian besar data yang ada, namun tipe dasar tidak cukup untuk memproses banyak data, apalagi data yang memiliki struktur tertentu. Untuk menyelesaikan persoalan pengolahan data tertentu, suatu tipe data baru dapat dibentuk berdasarkan tipe data dasar (primitif).

Tipe Enum

Tipe enum adalah suatu tipe yang elemennya didefinisikan sendiri satu per satu. Dalam representasinya sebenarnya tipe enum ini adalah sebuah integer yang diberi nama. Dalam Pascal tipe ini didefinisikan dengan cara menyebut elemen-elemennya:

```
type
    hari = (senin, selasa, rabu, kams, jumat, sabtu, minggu);
    warna = (merah, kuning hijau);
```

Tipe enum tidak bisa dibaca (dengan readln) atau ditulis (dengan writeln), tipe jenis ini hanya bisa diberi nilai dengan assignment.

Tipe Enumerasi

Tipe enumerasi adalah tipe yang elemen-elemennya bisa disebutkan satu persatu (bisa dicacah), integer, enum, dan karakter adalah contoh tipe enumerasi. Tipe real tidak bisa dicacah satu persatu, tipe string juga tidak bisa dicacah satu per satu.

Subtipe Integer

Integer memiliki range tertentu sesuai dengan jumlah bit yang dipakai oleh integer. Terkadang dalam kasus tertentu hanya diperlukan subrange (sebagian range) integer, misalnya untuk mengolah data

jam yang berbasis 60 (seksadesimal), yang diperlukan hanyalah angka dari 0 sampai 59, angka di luar itu sifatnya tidak valid. Subtipe integer didefinisikan dengan menyebutkan range untuk tipe tersebut.

```
Type jam = 1..12;
      menit = 0..59;
      detik = 0..59;
```

Pengecekan *run time* dan *compile time*

Perhatikan bahwa jika Anda memiliki variabel `m` dengan tipe menit, lalu melakukan hal ini:

```
m = 61;
```

maka kompilator akan menolak program karena ada pemeriksaan pada waktu kompilasi yang mencegah `m` diisi selain 0 sampai 59. Namun jika dalam program dilakukan hal ini:

```
readln(m);
```

maka kompilator tidak akan menolak jika pengguna memasukkan angka selain 0 sampai 59, dengan kata lain, kompilator hanya melakukan pengecekan waktu kompilasi (*compile time*), tapi tidak waktu eksekusi (*run time*).

Cara yang benar untuk membaca tipe menit agar masuk ke `m` adalah dengan membaca integer ke dalam variabel lain dan memeriksa hasil pembacaan, seperti ini:

```
var i: integer;
    m: menit;
begin
    repeat
        readln(i);
    until (i >= 0) and (i <= 59);
    m := i; (* bilangan yang dimasukkan ke m pasti sudah valid *)
end.
```

Tipe SET (himpunan)

Tipe himpunan adalah tipe yang bisa menerima himpunan nilai yang masing-masing elemennya adalah tipe enumerasi. Perhatikan: tidak semua bahasa pemrograman prosedural memiliki tipe SET. Deklarasi tipe himpunan adalah:

```
type
    hari = (senin, selasa, rabu, kamis, jumat, sabtu, minggu);
    setkar = set of char;
    harihari = set of hari;
```

Operasi yang tersedia untuk himpunan meliputi: gabungan (union), irisan (intersection), dan pengurangan elemen himpunan, serta pengecekan keanggotaan.

Tipe set tidak bisa dibaca dan ditulis secara langsung menggunakan `read/readln/write/writeln`.

Tipe Komposisi (Record)

Suatu tipe bisa disusun dari beberapa tipe, misalnya tipe mahasiswa bisa disusun dari tipe string untuk nama, tipe real untuk nilai, dan tipe integer untuk nomor urutnya. Deklarasi tipe komposisi dalam Pascal adalah:

```
type mahasiswa = record
    nama: string;
    urut: integer;
    nilai: real;
end;
var mhs: mahasiswa;
```

Cara mengakses elemen tipe adalah dengan titik, misalnya:

```
writeln(mhs.nama);
```

Atau dengan blok with :

```
with mhs do
begin
    writeln(nama);
end;
```

Jika bagian dari tipe bentukan merupakan tipe dasar yang bisa langsung dibaca atau tulis maka elemen tersebut bisa langsung dibaca atau ditulis, namun bagian tipe yang tidak bisa dibaca dan ditulis langsung tetap harus diperlakukan khusus.

Tabel Berdimensi Satu (Array)

Jenis variabel yang telah diberikan hanya bisa digunakan untuk menyimpan sebuah nilai saja. Dalam banyak kasus kita perlu menyimpan banyak nilai yang serupa untuk diproses, misalnya data nilai mahasiswa dalam suatu kelas untuk dihitung rata-ratanya.

Tabel adalah tipe data yang dapat menampung sejumlah data dengan tipe sejenis, jumlah data yang dapat disimpan dibatasi oleh kemampuan kompilator dan komputer. Deklarasi tabel integer yang terdiri dari 100 elemen adalah:

```
var
    tabint : array [1..100] of integer;
```

Dengan deklarasi semacam itu sebuah tabel yang terdiri dari 100 elemen integer dibentuk, dan dapat diakses melalui indeksinya (antara 1 sampai 100, inklusif). Untuk mengakses elemen tabel ke-n gunakan sintaks: tabint[n].

Tabel dapat diproses menggunakan loop (biasanya loop for, karena indeks tabel sudah jelas), contoh berikut akan menjumlahkan seluruh elemen tabel integer yang dideklarasikan di atas (jumlah dan i bertipe integer):

```
jumlah:=0;
for i:=1 to 100 do jumlah:=jumlah+tabint[i];
writeln('Jumlah elemen tabel adalah:', jumlah);
```

String sebagai Array of Character

String sebenarnya adalah tabel berdimensi satu dengan elemennya berupa karakter, indeks ke-0 tabel berisi panjang string saat ini, dan indeks ke 1 dan seterusnya berisi data karakter yang ada pada string. Pada string seperti ini:

```
s := 'hello';
```

maka s[1] = 'h', s[2]='e', dst. Sedangkan ord(s[0]) akan berisi panjang string yaitu 5. Pengaksesan panjang string melalui elemen ke-0 tidak disarankan, karena tergantung pada implementasi Pascal, elemen string sebaiknya hanya diakses mulai dari elemen 1 sampai panjang string.

Tabel Berindeks Banyak (Tabel Multi Dimensi)

Terkadang kita perlu memiliki tabel dengan dimensi lebih dari satu. Matriks merupakan salah satu contoh tabel dengan banyak dimensi (tabel multi dimensi). Tabel multi dimensi dipandang oleh semua bahasa yang mengenal tipe data tabel satu dimensi, karena tabel dua dimensi bisa dipandang sebagai tabel dari tabel.

Dalam Pascal, tipe tabel multi dimensi dapat dideklarasikan seperti ini:

```
var  
  matriks : array [1..4, 1..4] of integer;
```

Pengaksesan elemen tabel dilakukan mirip seperti pengaksesan tabel satu dimensi:

```
matriks[baris, kolom] := nilai;
```

Pemrosesan tabel multi dimensi umumnya dilakukan dengan nested loop. Hal yang perlu diperhatikan dalam pemrosesan tabel dengan loop adalah bahwa indeks tabel tidak boleh lebih dari yang sudah dideklarasikan.

Bab 9

FILE

File adalah suatu bentuk penyimpanan eksternal dalam suatu media penyimpanan. Program yang ditulis adalah sebuah file, hasil kompilasi (program) juga adalah sebuah file. Ketika Anda mengedit dengan editor, yang Anda edit merupakan file yang jenisnya adalah teks.

File berguna untuk menyimpan data yang akan dipakai kembali, karena apa yang disimpan di file akan ada di media penyimpanan sekunder (harddisk atau disket), dan dapat dipakai kembali meskipun komputer sudah dimatikan. Hal ini berbeda dengan variabel yang nilainya akan hilang ketika program selesai berjalan.

File teks dan File biner

File teks adalah file yang isinya bisa dibaca langsung dengan editor teks biasa, sedangkan file biner adalah file yang memiliki format khusus yang harus dibaca oleh program khusus. Dalam lomba TOKI dan IOI, jenis file yang digunakan hanya file Teks, sehingga buku ini juga hanya membahas penggunaan file teks.

Nama Fisik File

File di dalam disket atau harddisk (atau media penyimpanan manapun) akan diidentifikasi dengan menggunakan suatu nama file. Nama ini disebut dengan nama fisik. Sistem operasi memiliki batasan dalam penamaan file ini (misalnya di DOS atau Windows nama file tidak boleh mengandung karakter titik dua, di DOS nama file maksimal 11 Karakter).

Setiap sistem operasi juga memiliki sifat yang berbeda dalam menangani perbedaan case pada nama file. Windows dan DOS menganggap Aku dan AKU adalah file yang sama, sementara Linux menganggap bahwa Aku dan AKU mengacu ke dua file yang berbeda. Perbedaan dalam sistem operasi ini perlu diperhatikan oleh program agar dapat berjalan di semua lingkungan.

Primitif Pemrosesan File

Ada lima primitif yang digunakan dalam memproses file, yang meliputi assign nama fisik ke nama logik, pembukaan file, pembacaan file, penulisan file, dan penutupan file.

? Assign nama fisik ke nama logik

Pemrosesan file diawali dengan assignment nama fisik (berupa string yang merupakan nama yang dikenali sistem operasi) ke nama logik (suatu variabel dalam program yang memiliki tipe file atau tipe text). Proses assignment ini dilakukan agar dalam semua langkah berikutnya file bisa diacu berdasarkan variabel tersebut dan bukan nama fisiknya. Assignment nama fisik ke nama logik tidak melakukan hal apapun terhadap file.

Dalam beberapa bahasa, proses assignment dari nama fisik ke nama logik dilakukan ketika file dibuka. Dalam Pascal, assignment ini merupakan langkah yang terpisah dari langkah membuka file. Contoh assign:

```
var t: text;  
begin  
    assign(t, 'file.txt');  
end.
```

? Membuka file

Setelah proses assignment dilakukan, maka file dapat dibuka. Ketika membuka file, harus ditentukan operasi apa yang hendak dilakukan terhadap file tersebut, apakah file akan dibaca saja, atau file akan ditulisi, atau file akan ditambah isinya di bagian akhir (append), atau file akan dibaca dan ditulisi.

Pada Pascal, file teks bisa dibuka untuk dibaca dengan memanggil prosedur reset, dibuka untuk ditulisi (isi file sebelumnya akan dihapus) dengan prosedur rewrite, dan dibuka untuk ditambah isinya dengan prosedur append. File teks di Pascal tidak bisa dibuka untuk dibaca dan ditulisi sekaligus.

? Membaca file

File yang dibuka dengan prosedur reset bisa dibaca menggunakan read atau readln. Proses pembacaan seolah-olah seperti ada pengguna yang mengetikkan suatu input melalui keyboard. Jika ada file dengan nama input.txt dengan isi berikut:

```
hello  
123
```

(isi file satu baris kata hello, diikuti satu baris angka 123). maka jika program berikut ini dijalankan:

```
var t: text;
    s: string;
    i: integer;
begin
  assign(t, 'input.txt');
  reset(t); (*buka file untuk dibaca*)
  readln(t, s); (*baca string dari file*)
  readln(t, i); (*baca integer dari file *)
  writeln(s); (*tuliskan string yang dibaca dari file *)
  writeln(i); (*tuliskan integer yang dibaca dari file *)
  close(t);
end.
```

Maka hasilnya adalah program akan mencetak hello dan diikuti dengan angka 123 di baris berikutnya. Kata hello dan angka 123 dibaca dari file input.txt.

? Menulis file

Sebaliknya, sebuah file yang dibuka dengan menggunakan append bisa ditulisi dengan menggunakan write atau writeln, dengan parameter pertama adalah nama logik file. Penulisan dengan write dan writeln yang dilakukan terhadap file akan tertulis pada file dan tidak pada layar. File yang sudah ditulisi ini bisa dibaca lagi dengan program yang serupa.

Program berikut ini akan menuliskan 2 baris ke dalam sebuah file teks, baris pertama adalah kata hello dan baris kedua adalah sebuah integer 123

```
var t: text;
begin
  assign(t, 'output.txt');
  rewrite(t); (*buka file untuk ditulisi*)
  writeln(t, 'Hello'); (*tuliskan string ke dalam file*)
  writeln(t, 123); (*tuliskan integer ke dalam file *)
  close(t);
end.
```

Hasil output program di atas dapat dibaca oleh contoh program sebelumnya.

? Menutup file

Setelah selesai, maka file perlu ditutup. Setelah file ditutup sistem operasi akan memastikan bahwa file dipindahkan ke media penyimpanan, sebelum file ditutup, file mungkin belum dituliskan ke media penyimpanan. Penutupan file dalam Pascal dilakukan dengan memanggil prosedur close dengan parameter nama logik file yang telah dibuka.

Skema dasar Pembacaan File

Semua soal TOKI dan IOI mewajibkan peserta membaca input dari sebuah file dan menuliskan hasil ke dalam file lain. Setiap soal TOKI selalu diberi jumlah baris yang akan dibaca berikutnya, misalnya file berisi :

```
5
adi
budi
charlie
desi
eni
```

Skema pembacaan yang dilakukan untuk membaca soal semacam itu adalah dengan loop for dengan terlebih dahulu membaca jumlah yang akan diloop, misalnya seperti ini (anggap t adalah nama logik untuk file yang sudah dibuka untuk dibaca, i adalah sebuah integer, dan tabs adalah array of string):

```
readln(t, jumlahbaris);
for i:=1 to jumlahbaris do
begin
  readln(t, tabs[i]);
end;
```

Pembacaan file juga bisa dilakukan terhadap file yang tidak diketahui jumlah barisnya dengan menggunakan fungsi eof. Fungsi eof akan mengembalikan true jika akhir dari file telah tercapai, dan false jika belum.

```
(* buka file *)
While (not eof(t)) do
begin
  readln(t, s);
  (* lakukan pemrosesan *)
end;
```

STDIN dan STDOUT

Sebenarnya fungsi read/readln dan write/writeln tanpa parameter file akan menganggap dua buah nama logik untuk input dan output. Read dan Readln akan membaca dari nama logik STDIN yang merupakan file input standar (yang dipetakan oleh sistem operasi menjadi input dari keyboard), sedangkan Write dan Writeln akan menulis ke nama logik STDOUT yang merupakan file output standar (yang dipetakan oleh sistem operasi menjadi output ke layar komputer). Sehingga instruksi

```
readln(s);
```

sama saja dengan

```
readln(stdIn, s);
```

dan

```
writeln(s);
```

sama saja dengan

```
writeln(stdout, s);
```

Anda mungkin akan banyak menemui contoh program dari peserta TOKI yang memakai teknik seperti ini:

```
assign(stdIn, 'input.txt');  
reset(stdIn);
```

yang memetakan ulang stdIn ke file 'input.txt' sehingga untuk membaca file cukup dilakukan dengan

```
readln(s); (* sama dengan readln(stdIn, s) *)
```

Sebaliknya stdout juga bisa dipetakan agar semua writeln akan muncul di file.

```
Assign(stdout, 'fileout.txt');  
rewrite(stdout);
```

Teknik semacam ini sebaiknya tidak digunakan, karena selain mengurangi keterbacaan program, program memakai fitur sistem operasi (yang kebetulan saat ini didukung oleh Windows dan Linux), yang mungkin tidak akan ada di semua sistem.

Bab 10

REKURENS

Fungsi dan prosedur bisa memiliki sifat yang rekursif, artinya prosedur dan fungsi tersebut memanggil dirinya sendiri dalam bagian implementasinya. Struktur data juga bisa memiliki sifat yang rekursif, yang artinya struktur data tersebut bisa menunjuk ke struktur data yang lain yang sama.

Struktur data rekursif tidak akan dibahas dalam buku ini, karena terlalu kompleks. Fungsi rekursif hanya akan dibahas perkenalannya saja. Konsep rekursif bisa sangat rumit, dan akan dibahas di buku lain yang mengajarkan problem solving dan algoritma.

Fungsi yang rekursif

Fungsi faktorial adalah contoh fungsi yang rekursif, fungsi rekursif ini mengandung dirinya sendiri dalam definisinya:

$$\text{faktorial } n = \text{faktorial } (n - 1) * n$$

dan

$$\text{faktorial } 0 = 1$$

Perhatikan bahwa ada dua definisi fungsi ini, yaitu untuk $n = 0$ (yang hasilnya adalah 1) dan untuk n lebih dari nol (hasilnya adalah faktorial $(n - 1)$ dikalikan dengan n). Bagian yang tidak memanggil dirinya sendiri (dalam kasus $n = 0$) disebut dengan basis, sedangkan bagian yang memanggil dirinya sendiri disebut sebagai bagian rekurens.

```
Function faktorial (n: integer): int;  
begin  
    if (n=0) then (*basis*)  
    begin  
        faktorial := 1;  
    end else (* rekurens *)  
    begin  
        faktorial := n * faktorial ( n - 1);  
    end;  
end;
```

Tanpa bagian basis, maka rekursi tidak akan berhenti, jadi bagian basis ini harus ada, dan harus dijamin bahwa pada suatu saat kondisi basis akan dipenuhi.

Rekursif dan iteratif

Suatu bentuk rekursif bisa diubah menjadi bentuk iteratif (bentuk perulangan/loop), misalnya fungsi faktorial di atas dapat diubah menjadi:

```
Function faktorial (n: integer): int;  
var i, hasil: integer;  
begin  
    i := 0;  
    hasil := 1;  
    while (i < n) do  
    begin  
        i := i + 1;  
        hasil := hasil * i;  
    end;  
    faktorial := hasil;  
end;
```

Secara umum semua bentuk rekursif bisa diubah menjadi bentuk iteratif.

Lampiran 1

Contoh Soal dan Pembahasan Ujian Teori Pemrograman Pascal

SOAL-SOAL DEKLARASI

1. Manakah yang mendeklarasikan tipe enumerasi dengan tepat?
 - a. Type a=integer;
 - b. Type a=1..300;
 - c. Type a=(baik, jelek, buruk);
 - d. Type a=[baik, jelek, buruk];
 - e. Type a=baik, jelek, buruk;

Jawab:

- c. Type a=(baik, jelek, buruk);
2. Tipe di bawah ini mana yang tidak dapat melakukan operasi aritmatika?
 - a. integer
 - b. byte
 - c. real
 - d. boolean
 - e. word

Jawab:

- d. boolean
3. Deklarasi prosedur manakah yang dibenarkan?
 - a. procedure hapus;
 - b. procedure hapus(s:string);
 - c. procedure hapus(var s:string);
 - d. procedure hapus(s:string):boolean;
 - e. procedure hapus(var data);

Jawab:

- d. procedure hapus(s:string):boolean;

Pembahasan:

Untuk penulisan prosedur, tidak diperbolehkan adanya nilai kembali. Sedangkan parameter tanpa tipe data (pada opsi e), dapat dibenarkan.

4. Deklarasi function manakah yang tidak diizinkan?
- Function density(x:real):real;
 - Function density(b:byte):byte;
 - Function density(var s:string):real;
 - Function density(var data):byte;
 - Function density;

Jawab:

e. Function density;

Pembahasan:

Untuk penulisan function , harus ada nilai kembali.

5. Tipe variabel ekspresi manakah yang tidak dapat ditampilkan dengan procedure Writeln?
- Type T=Integer;
 - Type T=String;
 - Type C=Char;
 - Type T=(Small, Medium, Large)
 - Semua valid

Jawab:

d. Type T=(Small, Medium, Large)

Pembahasan:

Opsi d adalah tipe data enumerasi. Tipe data enumerasi tidak dapat ditampilkan dengan perintah Writeln.

6. Dengan deklarasi berikut:
- ```
Type warna=(merah, kuni ng, hi j au, bi ru, hi tam, puti h, j i ngga);
Var w: warna;
```

Perintah mana yang salah?

- If w in [warna] then writeln('ada');
- w:=merah;  
w:=w + kuning;
- w:=[merah];
- w:=hijau;  
dec(w);
- w:='Merah';

**Jawab:**

d. `w:=hijau;`  
`dec(w);`

**Pembahasan:**

Variabel `w` adalah variabel yang mempunyai tipe data enumerasi yang merupakan salah satu tipe data ordinal. Karena merupakan tipe data ordinal, maka variabel `w` dapat dioperasikan dengan fungsi atau prosedur seperti `ORD`, `DEC`, `INC`, `PRED`, dan `SUCC`.

7. Pada deklarasi di atas, jika variabel `W1` berisi `[merah,kuning,hijau]` dan variabel `W2` berisi `[merah,kuning,hitam]` maka, jika diberikan statemen `W3 :=W1+W2`, `W3` akan berisi:
- `[merah,kuning,hijau,hitam]`
  - `[merah,kuning,hijau,merah,kuning,hitam]`
  - `[hijau,hitam]`
  - `[merah,kuning,merah,kuning,hijau, hitam]`
  - `[merah,kuning]`

**Jawab:**

a. `[merah,kuning,hijau,hitam]`

**Pembahasan:**

Operator `+` pada tipe data himpunan adalah gabungan atau union.

8. Jika diberikan statemen `W3 :=W1-W2`, `W3` akan berisi:
- `[merah,kuning,hijau,hitam]`
  - `[merah,kuning,hijau,merah,kuning,hitam]`
  - `[hijau]`
  - `[merah,kuning,merah,kuning,hijau, hitam]`
  - `[merah,kuning]`

**Jawab:**

c. `[hijau]`

**Pembahasan:**

Operator `-` pada tipe data himpunan adalah operator *difference*.

9. Jika diberikan statemen  $W3 := W1 * W2$ ,  $W3$  akan berisi:
- [merah,kuning,hijau,hitam]
  - [merah,kuning,hijau,merah,kuning,hitam]
  - [hijau,hitam]
  - [merah,kuning,merah,kuning,hijau, hitam]
  - [merah,kuning]

**Jawab:**

c. [hijau,hitam]

**Pembahasan:**

Operator \* pada tipe data himpunan adalah operator irisan.

### SOAL-SOAL INPUT / OUTPUT

10. Perhatikan potongan program berikut ini :

```
Begin
Writeln((10 shr 1) shl 2);
end.
```

Apa yang dihasilkan oleh program diatas...

- 18
- 19
- 20
- 21
- 22

**Jawab:**

c. 20

**Pembahasan:**

Operator SHR adalah operasi pergeseran bit ke kanan dan operasi shl adalah operasi pergeseran bit ke kiri.

$10 \text{ shr } 1 = 5$  (  $1010 \text{ shr } 1 = 0101 = 5$  )

$5 \text{ shl } 2 = 20$  (  $0101 \text{ shl } 2 = 10100 = 20$  )

**SOAL-SOAL STRUKTUR KONTROL**

11. Bagaimana keluaran program di bawah ini?

```
Var
 I: integer;
Begin
 I:=2;
 Case I of
 1, 3, 5, 7, 9: writeln(' Ganjil');
 2: writeln(' Prima genap');
 0..10: writeln(' Normal');
 else writeln(' Tidak normal');
 end;
end;
```

- a. Prima genap
- b. Normal
- c. Prima genap  
Normal
- d. Normal  
Prima genap
- e. Prima genap  
Tidak normal

**Jawab:**

- a. Prima genap

**Pembahasan:**

Struktur kendali case akan segera keluar untuk menjalankan statement berikutnya setelah menemukan nilai yang tepat.

Perhatikan program di bawah ini:

```
var I, j, k: integer;
 L: byte;
begin
 i:=3;
 j:=4;
 k:=32;
 L:=0;
 {If - 1 }
 if i + j and k =0 then
 writeln(' Betul')
 else
```

```
writeln(' Salah');
{If - 2 }
if (i = 2) and (j < i) or (k > i) then
 writeln(' Betul')
else
 writeln(' Salah');
{If - 3}
if not L in [1..120] then
 writeln(' Betul')
else
 writeln(' Salah');
end.
```

Program diatas berisi tiga perintah if then else yang saling tidak berkaitan, masing-masing IF diberi nama IF - 1, IF - 2, IF - 3.

12. Perintah if manakah yang tidak dibenarkan:

- If - 1
- If - 2
- If - 3
- If - 1 dan if - 2
- Tidak ada if yang salah

**Jawab:**

e. Tidak ada if yang salah

**Pembahasan:**

Pada If - 1, ekspresi  $i+j$  and  $k$  adalah ekspresi matematika dengan urutan pengerjaan  $j$  and  $k$  kemudian ditambahkan dengan  $i$ . Ini merupakan ekspresi yang valid dalam bahasa Pascal

Pada If - 3, ekspresi Not L akan dioperasikan terlebih dulu. Ini juga merupakan ekspresi yang valid dalam bahasa Pascal.

13. Pada program di atas, if mana yang menghasilkan output "Betul"?

- If - 1
- If - 2
- If - 3
- If - 1 dan if - 2
- Tidak ada if yang menghasilkan "Betul"

**Jawab:**

b. lf - 2

**Pembahasan:**

Urutan pengerjaan operator AND dan OR adalah AND akan dievaluasi terlebih dulu. Pada kondisi pertama, ( $i = 2$ ) and ( $j < i$ ) akan menghasilkan nilai FALSE, namun pada saat dievaluasi dengan menggunakan kondisi OR, yaitu ( $k > i$ ), akan menghasilkan TRUE, sehingga yang dicetak adalah "Betul"

**SOAL-SOAL PERULANGAN**

14. Perhatikan penggalan program berikut ni :

```
const
 Data: array [1..3, 1..3] of char=
 (('1', '1', '2'), ('2', '2', '4'), ('4', '4', '8'));
var i, j : byte;
begin
 for i:= 1 to 3 do
 begin
 for j:=3 downto 1 do
 write(Data[i,j]):
 writeln;
 end;
 end.
```

Apa keluaran program di atas ?

- a. 112  
224  
448
- b. '1' '1' '2'  
'2' '2' '4'  
'4' '4' '8'
- c. 211  
422  
844
- d. '2' '1' '1'  
'4' '2' '2'  
'8' '4' '4'

- e. 124
- 124
- 248

**Jawab:**

- c. 211
- 422
- 844

15. Perhatikan program dibawah ini :

```
type data=set of char;
var setchar: data;
 s: string;
 i: integer;
begin
 setchar:=[];
 readln(s);
 for i:=1 to length(s) do
 begin
 if not(s[i] in setchar) then
 begin
 setchar:=setchar+[s[i]];
 write(s[i]);
 end;
 end;
 writel n;
end.
```

Output dari program di atas jika input 'To be or Not To be that is the question' adalah

- a. 'To berNthaisqun.'
- b. 'To berNhaisqu`
- c. 'to@bernhaisquN'
- d. 'T N.'
- e. 'OBERTHAISQUN'

**Jawab:**

- a. 'To berNthaisqun.'

**Pembahasan:**

Yang perlu diperhatikan adalah bahwa tidak ada anggota yang sama dalam sebuah set (himpunan).

16. Gunakan program berikut untuk menjawab pertanyaan :

```

type data=set of byte;
var setint: data;
 i: integer;
begin
 setint:=[1];
 setint:=setint+[3];
 setint:=[5];
 for i:=1 to 5 do
 begin
 if (i in setint) then continue else setint:=[i];
 end;
 end.

```

Output dari program di atas adalah:

- [1,2,3,4,5]
- [1,3,5]
- [5]
- [1,3]
- []

**Jawab:**

- [5]

**Pembahasan:**

Statement di bawah ini

```

setint:=[1];
setint:=setint+[3];
setint:=[5];

```

Akan membuat setint berisi [5] saja. Pada statement berikutnya:

```

for i:=1 to 5 do
 begin
 if (i in setint) then continue else setint:=[i];
 end;

```

Akan membuat setint berisi nilai terakhir dari i yaitu 5.

### **SOAL-SOAL PROSEDUR DAN FUNGSI**

17. Perhatikan program berikut :

```

var s: string;

```

```
begin
s:=' TOKI GO GET GOLD!';
delete(s, 1, length(s) - 12);
writeln(s);
end.
```

Apa keluaran program di atas ?

- GO GET GOLD!
- GO GET GOLD!
- GET GOLD!
- TOKI GO GET
- TOKI GO GE

**Jawab:**

- GO GET GOLD!

**Pembahasan:**

Procedure delete:

Deklarasi : procedure Delete(var S: String; Index: Integer; Count: Integer);

Keterangan : procedure delete akan menghapus S sebanyak count karakter, dimulai dari posisi Index.

Function length:

Deklarasi : Function Length (S : String) : Integer;

Keterangan : Length menghasilkan panjang dari S, bernilai antara 0 sampai dengan 255. Jika S tidak berisi apa-apa maka akan menghasilkan 0.

Statement delete(s,1,length(s)-12) akan menghapus s dari posisi 1 sebanyak panjang s, yaitu  $17-12 = 5$ . Sehingga yang dihapus adalah karakter 'TOKI' dan s akan bernilai GO GET GOLD!

18. Perhatikan penggalan program berikut :

```
var i, k: integer;
begin
i:=5; k:=0;
k:=trunc(sqrt(i))+1;
writeln(k);
end.
```

Apa keluaran program di atas ?

- 3
- 2.24

- c. 2
- d. 0
- e. program tidak dapat dijalankan

**Jawab:**

- a. 3

**Pembahasan:**

Fungsi sqrt :

Deklarasi : Function Sqrt (X : Real) : Real;

Keterangan : menghasilkan akar pangkat dua dari x, di mana x harus positif

Fungsi trunc:

Deklarasi : Function Trunc (X : Real) : Longint;

Keterangan : menghasilkan bilangan bulat dari X, akan selalu lebih kecil atau sama dengan X.

Sqrt(5) akan menghasilkan 2.23

Trunc(2.23) akan menghasilkan 2

Sehingga k:=trunc(sqrt(i))+1; akan menghasilkan 3

19. Mengacu pada program berikut :

```
var
 A, B: string;
 C: string[10];

begin
 A:= ' TOKI MEMANG' ;
 B:= ' HEBAT' ;
 C:=A+B;
 if (Pos(B)>0) then
 Begin
 Writeln(' A');
 end else
 Writeln(' B');
end.
```

Apa yang terjadi jika program di atas di jalankan...

- a. Huruf 'A' tercetak
- b. Huruf 'B' tercetak
- c. Tidak dapat dipastikan
- d. Terjadi error
- e. Tidak bisa di compile

**Jawab:**

e. Tidak bisa di compile

**Pembahasan:**

Kesalahan pertama yang akan ditemui program adalah pada function pos.

Deklarasi : `Function Pos (Substr : String; S : String) : Integer;`

Keterangan : function pos akan menghasilkan urutan atau posisi substr di S. Jika tidak ditemukan, maka akan menghasilkan 0.

Pada program function pos hanya terdiri dari 1 parameter saja sehingga program tidak akan dapat dijalankan.

20. Perhatikan potongan program berikut :

```
begin
 writeln(round(frac(3.7)));
end.
```

Apa keluaran program di atas ?

a. 0      b. 1      c. 2      d. 3      e. 4

**Jawab:**

b. 1

**Pembahasan:**

Fungsi frac (lihat pembahasan di atas)

Fungsi round

Deklarasi : `Function Round (X : Real) : Longint;`

Keterangan : membulatkan bilangan X, yang mungkin lebih besar atau lebih kecil dari X.

Frac(3.7) akan menghasilkan 0.7

Round(0.7) akan menghasilkan 1

21. Diketahui deklarasi fungsi dan variabel sebagai berikut:

```
var St: String;
procedure Sulap(var S: String);
begin
 if S = 'Kecil' then S := 'kecil' else
```

```
if S = ' Besar' then S := ' BESAR' ;
end;
```

Di antara potongan program berikut, manakah yang salah?

- a. St := Chr(60);  
Sulap(St);
- b. St := 'KECIL';  
Sulap(St);
- c. St := Chr(45) + Chr(65);  
Sulap(St);
- d. Sulap('Besar');
- e. Semua ekspresi di atas benar

**Jawab:**

- a. St := Chr(60);  
Sulap(St);

**Pembahasan:**

Sebuah variabel string tidak dapat diberikan nilai bertipe data character.

### **SOAL-SOAL OPERASI FILE**

22. Perintah mana yang tidak boleh digunakan untuk file bertipe text?
- a. Assign
  - b. Reset
  - c. EOF
  - d. FilePos
  - e. Semua boleh digunakan untuk Text

**Jawab:**

- d. FilePos

**Pembahasan:**

Perintah FilePos adalah perintah untuk mengetahui posisi file pointer (penunjuk file), dan hanya dapat dioperasikan untuk file bertipe bukan text.

Gunakan program berikut ini untuk menjawab soal di bawah ini:

```
program Uji;
var T: Text;
 i, j, k: integer;
begin
 Assign(T, 'INPUT.TXT');
 Reset(T);
 Readln(T, i, j, k);
 Writeln(i, ' ', j, ' ', k);
 Readln(T, i);
 Readln(T, j);
 Writeln(i, ' ', j);
 Close(T);
End.
```

23. Misalkan file INPUT.TXT berisi baris-baris sebagai berikut:

```
3 1 4 9
5 2 6
8 7
0
```

Bagaimanakah output dari program tersebut?

- a. 3 1 4 9  
5 2 6  
8 7
- b. 3 1 4  
9 5
- c. 3 1 4  
5 2
- d. 3 1 4  
5 8
- e. Terjadi runtime error karena isi file INPUT.TXT tidak sesuai untuk program ini.

**Jawab:**

- d. 3 1 4  
5 8

**Pembahasan:**

Perintah Readln akan melakukan pembacaan di baris berikutnya. Perintah Readln pertama akan melakukan pembacaan pada file baris pertama, perintah Readln berikutnya melakukan pembacaan pada baris ke dua dan perintah Readln terakhir melakukan pembacaan pada baris ketiga.

**SOAL-SOAL KASUS / MEMBACA PROGRAM**

Program berikut ini dipakai untuk menjawab dua soal di bawah ini

```

var Bil : Integer;

procedure Find(B: Integer; I: Integer);
var J, R: Integer;
begin
 R:=Round(sqrt(B));
 J:=2;
 while (J<=R) and (B Mod J<>0) do
 inc(J);
 if J<=R then
 begin
 Write(J, '*');
 Find(B div J, I+1);
 end
 else if I>0 then
 Writeln(B, '=', Bil)
 else
 Writeln(' Bilangan Prima!');
end;

begin
 Write(' Masukkan bilangan : ');
 Readln(Bil);
 Find(Bil, 0);
end.

```

24. Bagaimana output program di atas bila inputnya 42?
- $7 * 3 * 2 = 42$
  - Bilangan prima
  - $=42$
  - $2 * 3 * 7 = 42$
  - Salah semua

**Jawab:**

a.  $7 * 3 * 2 = 42$

25. Bagaimana output program di atas bila, inputnya 23?

- a. = 23
- b. Bilangan prima
- c.  $23 * 1 = 23$
- d. = 23 Bilangan prima!
- e. Salah semua

**Jawab:**

- b. Bilangan prima

Joni, petugas statistik yang baru saja belajar Pascal. Mencoba membuat program perata-rata sebagai berikut

```
var Amatan: array[5] of integer;
 Jumlah: Integer;
 RataRata: Integer;
 I: Integer;

begin
 for I:=1 to 5 do
 begin
 Write(' Amatan ke-', I, ' : ');
 Readln(Amatan[I]);
 end;
 Jumlah:=0;
 for I:=1 to 5 do
 begin
 Jumlah:=Jumlah+Amatan[I];
 RataRata:=Jumlah/5;
 Writeln(' Jumlah = ', Jumlah);
 Writeln(' Rata-rata = ', RataRata);
 Readln;
 end.
end.
```

Gunakan program yang dibuat oleh Joni ini untuk menjawab soal-soal berikut.

26. Ketika si Joni mencoba menjalankan program tersebut, ternyata, compiler menunjukkan sebuah pesan kesalahan yang membuat: ia kebingungan. Tahukah Anda kesalahan pertama yang dibuat Joni?

- a. Judul program (`program Statistik`) terlalu panjang, maksimum 8 karakter (misalnya: `program Stat`)

- b. Procedure `Readln` (pada baris terakhir program sebelum `end.`) tidak boleh dipanggil tanpa parameter. Jadi seharusnya: `Readln(I);`
- c. Statement `for` dengan variabel sama tidak boleh diulangi dua kali. Seharusnya dideklarasikan variabel lain, misalnya `var I: Integer` untuk `for` yang kedua
- d. Deklarasi array salah, semestinya: `var Amatan: array[ 1..5] of Integer;`
- e. Nama variabel seperti `RataRata` tidak valid, seharusnya `Ratarata`

**Jawab:**

- a. Deklarasi array salah, semestinya: `var Amatan: array[ 1..5] of Integer;`

**Pembahasan:**

Deklarasi dari array adalah:

```
type
 identifier=array[tipe_indeks] of tipe_data
```

di mana `tipe_indeks` adalah tipe data ordinal.

27. Setelah Anda memberi saran demikian, ternyata Joni masih belum bisa meng-compile programnya. Apa sebabnya?
  - a. setiap variabel harus dideklarasikan dengan key word `var` sendiri-sendiri.  
Misalnya:  

```
var Jumlah: Integer;
var RataRata: Integer;
Var I: Integer;
```
  - b. Variabel `RataRata` tidak harus bertipe `Real`
  - c. Semua variabel, kecuali `I` seharusnya adalah `Real`, tidak boleh `Integer`
  - d. Pemisah antara parameter dalam `Write` dan `Writeln` harus titik koma, bukan koma, Misalnya  

```
Writeln('Jumlah = ';Jumlah);
```
  - e. Semua alasan di atas salah

**Jawab:**

- c. Semua variabel, kecuali `I` seharusnya adalah `Real`, tidak boleh `Integer`

**Pembahasan:**

Dalam program diberikan instruksi `RataRata:=Jumlah/5` yang berarti variabel `RataRata` harus bertipe `Real`. Karena operator `/` hanya dikenal oleh variabel yang bertipe `real`.

28. Joni mengganti operator “/” dengan "div" pada baris ke-15 program tersebut. Apa akibatnya?
- program tidak mau di-compile karena. operator div tidak dapat digunakan di situ
  - nilai rata-ratanya menjadi 5
  - nilai rata-ratanya menjadi 6
  - nilai rata-ratanya menjadi 0
  - nilai rata-ratanya menjadi 2

**Jawab:**

- b. nilai rata-ratanya menjadi 5

**Pembahasan:**

Perintah div adalah operator pembagian yang menghasilkan pembulatan ke bawah.

Gunakan program berikut ini untuk menjawab beberapa soal selanjutnya:

```
uses crt;

var j: array['A'..'Z'] of Byte;
 c: char;
 Kal: string;

procedure HH(S: String);
var i: integer; {baris-6}
 m: char;
begin
 for i:= 1 to length(S) do
 begin
 m:=S[i]; {baris-11}
 if m in ['A'..'Z'] then {baris-12}
 inc(J[i]);
 end;
end;

begin
 for c:='A' to 'Z' do J[c]:=0;
 Kal:='PASAR';
 HH(Kal);
 for c:='A' to 'Z' do
 if J[c]>0 then write(c, J[c], ' ');
 writeln;
 Kal:='RAYA';
 HH(Kal);
 for c:='Z' downto 'A' do
 if j[c]>0 then write(c, J[c], ' ');
 writeln;
```

**end.**

29. Bila terdapat kesalahan yang menyebabkan program sama sekali tidak dapat dijalankan sebutkan pada baris berapa, dan bagaimana perbaikannya?
- Kesalahan semacam ini tidak ada
  - Baris 12, seharusnya ditulis  
`If [m] in ['A'..'Z'] then`
  - Baris 6 seharusnya ditulis  
`var i: Char;`
  - Baris 13, seharusnya ditulis  
`inc(J[m]);`
  - Index array hanya boleh berupa angka. Jadi deklarasi variabel seharusnya ditulis:  
`const A = 1; Z = 26;  
var J: array[A..Z] of Byte;  
    c: Byte;  
    kal: String;`

dan semua konstanta karakter dalam perintah for harus diganti, misalnya: for c:= A to Z do dan seterusnya

**Jawab:**

- b. Baris 13, seharusnya ditulis  
`inc(J[m]);`

**Pembahasan:**

Variabel J adalah variabel dengan tipe data array yang mempunyai indeks ['A'..'Z']. Dalam program diberikan indeks berupa bilangan bulat, yaitu i. Hal ini akan menghasilkan pesan kesalahan type mismatch.

30. Dengan perbaikan seperti nomor sebelumnya (kalau ada), maka program bisa dijalankan. Apakah hasil dari program tersebut?
- A2 P1 R1 S1  
A4 P1 R2 S1 Y1
  - A2 P1 R1 S1  
Y1 R1 A2
  - A2 P1 R1 S1  
Y1 R2 A4
  - P1 A2 S1 R1  
Y1 A4 R2
  - A2 P1 R1 S1

YI SI R2 PI A4

**Jawab:**

e. A2 P1 RI SI  
YI SI R2 PI A4

31. Tindakan apakah yang dilakukan oleh subrutin HH ketika dipanggil oleh baris 19 program di atas, dengan string S berisi kata "PASAR"?
- Menghitung frekuensi kemunculan huruf-huruf alfabet dan menyimpannya dalam array J
  - Mengumpulkan huruf-huruf alfabet yang muncul lebih dari satu kali ke dalam array J
  - Mencatat letak setiap huruf alfabet ke dalam array J
  - Menentukan huruf yang paling sering dan paling jarang muncul dalam array J
  - Mengurutkan huruf-huruf menurut urutan alfabet dari yang terkecil sampai yang terbesar.

**Jawab:**

a. menghitung frekuensi kemunculan huruf-huruf alfabet dan menyimpannya dalam array J



## **Lampiran 2**

### **Contoh Soal dan Pembahasan Ujian Praktek Pemrograman Pascal**

Contoh 1.

## **Cetak Angka** *(Soal ini pernah diberikan pada Practice Session OSN 2003, Balikpapan)*

Ketentuan umum :

**Nama Program** : CETAK.PAS  
**Batas Run-time** : 1 detik / test case  
**Nama File Masukan** : CETAK.IN  
**Nama File Keluaran** : CETAK.OUT

Tulis sebuah program yang membaca sebuah bilangan bulat  $n$  dan mencetak bilangan bulat dari 1 sampai dengan  $n^2$ .

### **FORMAT MASUKAN (Nama File: CETAK.IN)**

Masukan terdiri dari sebuah bilangan bulat  $n$  ( $1 \leq n \leq 10$ ).

### **CONTOH MASUKAN**

3

### **FORMAT KELUARAN (Nama File: CETAK.OUT)**

Keluaran terdiri dari  $n$  baris. Baris ke- $i$  ( $1 \leq i \leq n$ ) berisi  $(2i - 1)$  bilangan bulat, dengan tiap bilangan bulat dipisahkan oleh sebuah spasi. Setiap bilangan bulat terurut menurun sehingga setiap bilangan bulat yang tercetak selalu lebih besar daripada bilangan bulat di sebelah kanannya (bila ada) dan selalu lebih besar daripada bilangan-bilangan bulat pada baris-baris di bawahnya (bila ada).

### **CONTOH KELUARAN**

```
9
8 7 6
5 4 3 2 1
```

**PEMBAHASAN DAN SOLUSI**

Oleh : Ilham Winata Kurnia, Kontestan IOI 2002 Yong -In, Korea

Dalam soal ini, kita diminta untuk menampilkan bilangan 1 sampai dengan  $N^2$  secara terurut menurun. Selain itu, setiap barisnya, kita perlu menampilkan sejumlah bilangan (bila diperhatikan, pada baris ke- $i$ , ada  $2 * i - 1$  bilangan yang ditampilkan, dan ini adalah barisan bilangan ganjil). Jadi, kita cukup lakukan iterasi hingga ada tepat  $2 * i - 1$  bilangan yang tercetak pada baris ke- $i$  (dengan menggunakan perintah `write()`). Setelah itu baru kita cetak pengganti barisnya (dengan menggunakan perintah `writeln()`).

**Solusi :**

```
program CetakAngka(input, output);
Const InFile = 'CETAK.IN';
 OutFile = 'CETAK.OUT';
 MAXN = 10;
var i, j, k, n : integer;
begin
assign(input, InFile);
assign(output, OutFile);
reset(input);
rewrite(output);
readln(n);
j := n * n; k := j;
for i := 1 to n do
begin
write(j); dec(j);
while k - j >= (2 * i) - 1 do begin write(' ', j); dec(j); end;
writeln;
end;
close(input);
close(output);
end.
```

Contoh 2

## Menghitung Perulangan

Ketentuan umum :

**Nama Program** : HITUNG.PAS  
**Batas *Run-time*** : 1 detik / test case  
**Nama File Masukan** : HITUNG.IN  
**Nama File Keluaran** : HITUNG.OUT

Diberikan dua buah untaian huruf (*string*), hitung berapa kali string kedua muncul sebagai bagian dari string pertama. Asumsikan bahwa tiap kemunculan dari string kedua pada string pertama boleh saling menimpa (*overlap*). Panjang string pertama maksimal 10000, sementara panjang string kedua maksimal 200. String didefinisikan sebagai untaian karakter-karakter dengan kode ASCII 32 – 127 yang dibatasi oleh karakter-karakter dengan kode ASCII yang tidak termasuk dalam jangkauan 32– 127 tersebut.

### **FORMAT MASUKAN (Nama File: HITUNG.IN)**

Masukan terdiri dari dua baris. Baris pertama berisi string pertama, sementara baris kedua berisi string kedua.

### **CONTOH MASUKAN**

```
abcdefghijklmnlabcw
abc
```

### **FORMAT KELUARAN (Nama File: HITUNG.OUT)**

Keluaran hanya terdiri dari sebuah baris berisi sebuah bilangan bulat yang menyatakan banyak kemunculan string kedua pada string pertama.

### **CONTOH KELUARAN**

5

## PEMBAHASAN

**Oleh : Ilham Winata Kurnia, Kontestan IOI 2002, Yong In – Korea**

Inti dari “Menghitung Perulangan” tidak lain adalah berulang kali menyimulasikan perintah *find*, yang kita sering temui pada *software-software word processing* seperti Notepad, Edit, Star Office, dan sebagainya. Ada beberapa cara yang dapat digunakan untuk melakukan hal ini. Akan tetapi, untuk tingkat ini, hanya satu cara yang akan dibahas di sini, yaitu dengan *Brute Force*.

Permasalahan ini bisa dibagi menjadi 2 bagian: membaca input dan mencari keberadaan substring. Ada satu hal yang menyebabkan membaca input menjadi bagian tersendiri, yaitu panjang string pertama melebihi 255. Oleh sebab itu, kita tidak dapat serta merta melakukan pembacaan dengan menggunakan `readln` (walaupun nanti kita akan lihat sebuah pengecualian).

Salah satu cara untuk mengatasi hal ini adalah dengan mendefinisikan sebuah tipe variabel sendiri yang berupa `array[1..10000] of char`. Untuk membaca sebuah baris pada masukan, maka kita baca karakter demi karakter sampai kita temui karakter penanda akhir baris. Di Linux, karakter penanda akhir baris adalah ASCII #10 alias *newline character*, sementara pada Windows, karakter penanda akhir baris adalah rentetan ASCII #13 atau *linefeed character* dan ASCII #10.

Seperti yang dikatakan di atas, ada beberapa cara yang dapat dilakukan untuk menentukan keberadaan substring pada sebuah string, tapi kita akan hanya membahas satu saja. Cara *brute force* mengiterasi semua karakter pada string dan membandingkan setiap karakter pada bagian string yang sedang diiterasi dengan karakter-karakter dari substring. Bila setiap perbandingannya adalah benar, maka substring tersebut ada bagian dari string. Kita hanya cukup menghitung berapa kali perbandingan tersebut benar untuk mengetahui berapa kali kemunculan substring pada string tersebut.

## SOLUSI

```
program MenghitungPerulangan(input, output);
```

```
Const InFile = 'HITUNG.IN';
 OutFile = 'HITUNG.OUT';
 MAXA = 10000;
 MAXB = 200;
```

```
var a : array[1..MAXA + 1] of char;
 b : array[1..MAXB + 1] of char;
 cta, ctb, ans : integer;
 i, j : integer;
 cek : boolean;
```

```

begin
assign(input, InFile);
assign(output, OutFile);
reset(input);
rewrite(output);

cta := 0; ctb := 1; ans := 0;

while not eoln do
begin
inc(cta);
read(a[cta]);
end;

{ read input }
b[ctb] := #13;
while (b[ctb] = #10) or (b[ctb] = #13) do read(b[ctb]);
while not eoln and not eof do
begin
inc(ctb);
read(b[ctb]);
end;

{ find substring locations }
for i := 1 to cta - ctb + 1 do
begin
cek := true;
for j := 1 to ctb do
if a[i + j - 1] <> b[j] then begin cek := false; break; end;
if cek then inc(ans);
end;

writeln(ans);

close(input);
close(output);
end.

```

Adapun solusi lain yang lebih mudah adalah untuk menggunakan tipe `ANSIString` yang diberikan oleh Free Pascal. Dengan menggunakan tipe ini, kita tinggal menggunakan perintah `readln`, `writeln`, `copy`, dan `pos`. Hal ini dimungkinkan karena `ANSIString` dapat menyimpan untaian karakter dengan jumlah karakter hampir tak terhingga, tapi tetap mempertahankan sifat-sifat yang dimiliki oleh tipe variabel `string`. Dengan demikian, kita cukup melakukan satu iterasi saja, yaitu iterasi untuk mencari posisi dimana ada kemunculan substring pada string. Walaupun demikian, waktu yang dibutuhkan oleh program ini lebih kurang sama dengan program di atas karena perintah `pos` sendiri melakukan iterasi untuk mencari posisi substring. Di bawah ini adalah contoh source codenya:

```
program MenghitungPerulanganAlternatif(input, output);
```

```
var a, b : ANSIStrIng;
 la, i, j, ans : integer;

begin
assign(input, 'HITUNG.IN');
assign(output, 'HITUNG.OUT');
reset(input);
rewrite(output);

readln(a);
readln(b);
la := length(a);
i := 1; j := 1; ans := 0;

{ cari substring sebanyak-banyaknya }
while j <> 0 do
begin
j := pos(b, copy(a, i, la));
i := i + j;
if j <> 0 then inc(ans);
end;

writeln(ans);

close(input);
close(output);
end.
```

Seperti halnya soal “Cari Maksimum”, ada 10 test case yang digunakan. Berikut rinciannya:

| No. | Panjang String 1 | Panjang String 2 | Keterangan                                                                                    |
|-----|------------------|------------------|-----------------------------------------------------------------------------------------------|
| 1   | 1                | 1                | Tidak ada kemunculan                                                                          |
| 2   | 1                | 2                | Tidak ada kemunculan                                                                          |
| 3   | 495              | 1                | Mengetes pembacaan input, dengan string 2 adalah sebuah spasi                                 |
| 4   | 10000            | 2                | String 1 berisi hanya 3 jenis karakter                                                        |
| 5   | 100              | 3                | String 1 berisi hanya 3 jenis karakter                                                        |
| 6   | 500              | 5                | String 1 berisi hanya 2 jenis karakter, sementara string 2 berisi hanya sebuah jenis karakter |
| 7   | 1520             | 20               | String 1 berisi hanya 4 jenis karakter random dan disisipkan sebuah kemunculan dari string 2  |
| 8   | 10000            | 200              | Mengetes kasus terbesar. String 1 dan 2 hanya terdiri dari sebuah jenis karakter              |
| 9   | 4517             | 40               | Sama seperti 6, tapi string 2 berisi 2 buah jenis karakter                                    |
| 10  | 8188             | 100              | Sama seperti 1, tapi string 2 juga berisi 3 jenis karakter                                    |

Soal 3.

## Tempat Tidur Bebek

Nama File : TIDUR.PAS

Pak Dengklek punya  $N$  ( $1 \leq N \leq 2500$ ) bebek yang tidur di sebuah kandang besar dengan  $K$  kamar yang dinomori 0 sampai dengan  $K-1$ . Bebek ke- $i$  dinomori secara unik dengan nomor  $S_i$  ( $1 \leq S_i \leq 1000000$ ). Setiap bebek tahu di mana untuk tidur karena dia tidur di kamar nomor  $S_i \bmod K$ . Tentu saja, para bebek tidak mau membagi tempatnya untuk tidur.

Diberikan sebuah himpunan bebek dan nomornya, tentukan nilai minimum  $K$  sedemikian sehingga tidak ada 2 bebek yang tidur di kamar yang sama.

### FORMAT MASUKAN (TIDUR.IN)

Baris 1 : Sebuah bilangan bulat untuk  $N$

Baris 2.. $N+1$  : Sebuah bilangan bulat yang menyatakan nomor bebek

### CONTOH MASUKAN

5  
4  
6  
9  
10  
13

### FORMAT KELUARAN (TIDUR.OUT)

Sebuah baris dengan nilai minimum  $K$  pada baris tersebut.

### CONTOH KELUARAN

8

## PEMBAHASAN

**Oleh : Ilham Winata Kurnia, Anggota TOKI 2002, Yong-In - Korea Selatan**

Soal ini merupakan terjemahan dari soal berjudul “Cows in Beds” yang pernah dikeluarkan oleh United States of America Computing Olympiad (USACO) dalam salah satu kontesnya. Berikut adalah solusi dari salah satu pesertanya, Bruce Merry, yang berhasil melewati tes data yang dibuat oleh USACO (tes datanya lebih menantang daripada yang digunakan untuk seleksi TOKI).

Cara yang paling terlihat untuk menyelesaikan masalah ini adalah untuk melakukan loop melalui semua kemungkinan nilai  $K$  dan pada setiap iterasi, kita cari di mana setiap bebek tidur sambil menggunakan sebuah *array of boolean* untuk mengecek adanya perulangan (yaitu, ada dua bebek dalam sebuah kamar). Untuk mempercepat, kita dapat memulai iterasi dari  $N$  karena tidak mungkin kita dapat menempatkan  $N$  bebek di kurang dari  $N$  kamar tanpa adanya yang membagi tempat tidurnya (tampak jelas, tapi para matematikawan berpikir bahwa ini cukup penting untuk disebut *pigeon hole principle* atau prinsip burung dara).

Cara yang lebih ambisius adalah untuk menandai (dengan menggunakan sebuah *array of boolean*) semua nilai dari  $K$  yang tidak dapat digunakan dan carilah nilai pertama yang benar. Nilai-nilai  $K$  yang tidak dapat digunakan adalah faktor dari  $|S_i - S_j|$  untuk semua nilai  $(i, j)$ . Sayangnya, ada banyak sekali pasangan  $(i, j)$ , dan mencari faktor-faktor adalah sangat lambat. Mungkin kita bisa membuat beberapa trik yang memanfaatkan fakta bahwa ada **banyak** faktor yang diulang, bahkan nilai selisih yang berulang. Namun, dengan batas memori 16MB kita akan kesulitan untuk menyeimbangkan antara batas waktu dan memori, dan selain itu kita masih melakukan banyak sekali pembagian.

Pendekatan yang Bruce Merry gunakan adalah untuk menandai  $|S_i - S_j|$  sebagai nilai  $K$  yang tidak dapat digunakan, untuk setiap pasang  $(i, j)$ , lalu menggunakan algoritma yang pertama tapi dengan melewati nilai-nilai tersebut. Karena nilai dari  $|S_i - S_j|$  adalah dalam rentang  $0..1000000$ , maka memori bukan suatu masalah. Selain itu, karena tidak ada operasi pembagian atau perkalian, maka cara ini relatif cepat. Ide dasarnya adalah bahwa untuk tes data yang besar (tentu saja, tes-tes ini adalah tujuan dari optimasi kita) kita perlu menghilangkan sebanyak mungkin nilai  $K$  sehingga mengecek nilai-nilai yang tersisa dengan cara pertama adalah jauh lebih cepat.

Salah satu optimasi lain adalah untuk juga menandai nilai  $|S_i - S_j| / 2$  dimana  $|S_i - S_j|$  adalah bilangan genap. Ini menjamin bahwa semua nilai  $K$  antara 333334 dan 1000000 ditandai karena semua faktor lain dari  $|S_i - S_j|$

harus paling besar  $1000000/3=333333$ . Dengan kata lain, begitu nilai  $K > 333333$ , maka kita cukup melakukan sebuah tes penuh pada nilai  $K$  berikutnya yang tidak ditandai, karena nilai tersebut adalah jawabannya.

### SOURCE CODE (Oleh : Bruce Merry)

```

program cows_in_bed;
const
 inname = 'TIDUR.IN';
 outname = 'TIDUR.OUT';
var
 N : integer;
 K : longint;
 Si : array[1..5000] of longint;
 used : array[0..999999] of boolean;
 nogood : array[1..1000000] of boolean;
procedure readin;
var
 f : text;
 i : integer;
begin
 assign(f, inname);
 reset(f);
 readln(f, N);
 for i := 1 to N do
 readln(f, Si[i]);
 close(f);
end;
procedure makenogood;
var
 i, j : integer;
begin
 fillchar(nogood, sizeof(nogood), 0);
 for i := 1 to N - 1 do
 for j := i + 1 to N do
 nogood[abs(Si[i] - Si[j])] := true;
end;

procedure solve;
var
 i : integer;
 cur : longint;
 flag : boolean;
begin
 K := N - 1;
 flag := false;
 repeat
 inc(K);
 if nogood[K] then continue;
 flag := true;
 fillchar(used, sizeof(used[0]) * K, 0);
 for i := 1 to N do

```

```
begin
 cur := Si[i] mod K;
 if used[cur] then
 begin
 flag := false;
 break;
 end;
 used[cur] := true;
end;
until flag;
end;
procedure writeout;
var
 f : text;
begin
 assign(f, outname);
 rewrite(f);
 writeln(f, K);
 close(f);
end;
begin
 readin;
 makenogood;
 solve;
 writeout;
end.
```

Contoh 4.

## Spiral Huruf

Huruf-huruf yang disusun dalam konfigurasi di bawah ini adalah huruf alphabetis biasa. Tulislah program untuk mencetak spiral huruf dengan ukuran  $N$  ( $1 \leq N \leq 5$ , dan  $N$  ganjil) searah jarum jam. Spiral bisa dibuat dari tengah, atau dari salah satu pojok (tergantung dari inputnya). Data masukan adalah  $N$  dan  $A$ .  $A = 1$  artinya tengah,  $A = 2$  artinya pojok kiri atas,  $A = 3$  artinya pojok kanan atas,  $A = 4$  artinya pojok kanan bawah,  $A = 5$  artinya pojok kiri bawah.

### FORMAT MASUKAN

Data input disimpan dalam text file SPIRAL.IN. File ini berisi nilai  $N$  dan  $A$ .

### FORMAT KELUARAN

Simpan hasil tampilan spiral huruf pada file SPIRAL.OUT.

### CONTOH MASUKAN

3 4

### CONTOH KELUARAN

```
E F G
D I H
C B A
```

## PEMBAHASAN

Dalam masalah ini, kita diinginkan untuk membuat sebuah program yang dapat menghasilkan sebuah spiral huruf dalam bentuk mirip matriks  $N \times N$  dengan cara yang sudah ditentukan ( $A$ ). Pengertian spiral huruf dapat langsung dimengerti dengan melihat contoh keluaran. Ada sebuah frase kunci yang amat penting untuk menyelesaikan masalah ini, yaitu “searah jarum jam”.

Sebetulnya, soal ini tidak jelas karena tidak memberikan keterangan bagaimana spiral harus dibentuk bila  $A = 1$ . Ada 4 kemungkinan, yaitu bergerak ke atas, kiri, bawah dan kanan setelah mengisi kotak paling tengah.

Oleh karena itu, mari kita asumsikan bahwa setelah mengisi kotak yang ditengah maka kita bergerak ke sebelah kanan. Misal, untuk input 3 1, maka outputnya akan menjadi

```
G H I
F A B
E D C
```

### SOLUSI1

Bila kita melihat batas N, maka kita dapat simpulkan bahwa N hanya dapat bernilai 1, 3 dan 5. Karena untuk setiap N hanya ada 5 kemungkinan nilai A, maka total hanya ada 15 kemungkinan input. Mengingat jumlah yang begitu kecil ini, maka kita bisa mengerjakan semuanya secara manual, kemudian memasukkannya ke dalam program secara manual. Dengan menggunakan perintah if-then-else, langsung cetak solusinya yang telah kita masukkan secara manual ke dalam program.

### SOLUSI2

Cara yang lain untuk menyelesaikan masalah ini adalah untuk membuat program yang menyimulasikan pergerakan posisi huruf. Bayangkanlah output itu terletak pada suatu system koordinat kartesius. Posisi pojok kiri bawah memiliki koordinat (1, 1) dan posisi pojok kanan atas memiliki koordinat (N, N). Dalam contoh di atas, C berkoordinat (1, 1) dan G (3, 3). Kemudian, setiap huruf disimpan dalam sebuah tabel 2 dimensi dengan besar  $N \times N$ .

Kita misalkan koordinat-x dengan variabel  $x$  dan koordinat-y dengan variabel  $y$ . Bila kita ingin bergerak ke atas, maka  $x$  kita tambah dengan 1; turun,  $x$  kita kurang dengan 1; kanan,  $y$  kita tambah dengan 1; kiri,  $y$  kita kurang dengan 1. Karena dalam soal dikehendaki untuk mencetak spiralnya searah jarum jam, maka marilah kita buat aturan arah gerak sebagai berikut: kanan-bawah-kiri-atas-kanan-.... Yang sekarang jadi masalah adalah kapan kita harus berubah arah gerak.

Dengan mencoba-coba, maka kita dapat aturan sebagai berikut:

- Bila  $A = 1$  maka:

```
(x, y) = koordinat tengah tabel
Tabel[x, y] = "A"
Arah gerak = atas
```

BILA masih ada kotak kosong LAKUKAN  
    JIKA bisa bergerak dengan Arah gerak berikutnya MAKA  
        ubah Arah gerak menjadi Arah gerak berikutnya  
    JIKA TIDAK MAKA  
        bergerak sesuai dengan Arah gerak yang sekarang  
        Tabel[x, y] = huruf selanjutnya

- Bila A = 2 maka:

(x, y) = koordinat pojok kiri atas  
Tabel[x, y] = "A"  
Arah gerak = kanan

BILA masih ada kotak kosong LAKUKAN  
    JIKA bisa bergerak dengan Arah gerak saat ini MAKA  
        bergerak sesuai dengan Arah gerak saat ini  
        Tabel[x, y] = huruf selanjutnya  
    JIKA TIDAK MAKA  
        Arah gerak = Arah gerak berikutnya

- Langkah penyelesaian untuk A = 2 berlaku juga untuk A = 3, A = 4, dan A = 5. Hanya saja, posisi awal (x, y) diubah sesuai dengan yang tertulis di soal dan arah gerak pertama kali juga disesuaikan. Misal, untuk A = 5, maka posisi awal (x, y) adalah koordinat pojok kiri bawah dan Arah gerak = atas.

Contoh 5.

## Pembagian Panjang

Untuk setiap integer [bilangan bulat] yang diberikan, cetaklah hasil pembagian integer jika sebuah integer masukan dibagi dengan 13. Sisa pembagian tidak perlu di tulis. Jika hasil pembagian adalah 0, maka tuliskan saja nol sebagai outputnya.

### FORMAT MASUKAN

Terdiri atas sejumlah baris integer yang minimal terdiri atas 1 digit dan paling panjang 50 digit tanpa spasi sebelum dan sesudah integer di setiap barisnya .

### FORMAT KELUARAN

Hasil pembagian integer dari data input dengan 13.

### CONTOH

#### MASUKAN

0  
12  
13  
14  
25  
26  
262626  
1313131313131313131313131313

#### KELUARAN

0  
0  
1  
1  
1  
2  
20202  
10101010101010101010101010101

### PEMBAHASAN

Apa yang dituntut dari soal ini cukup jelas yaitu untuk mencetak hasil bulat dari suatu bilangan bulat yang dibagi dengan 13. Akan tetapi ada dua hal yang menjadi masalah. Pertama, panjang bilangan bulat bisa mencapai 50 digit. Kedua, tidak tertulis apakah input selalu merupakan bilangan bulat positif atau bisa juga

merupakan bilangan bulat negatif. Untuk masalah yang kedua, amat mudah untuk dipecahkan dengan 1 perintah `if`. Akan tetapi, untuk menyelesaikan masalah yang pertama butuh lebih dari itu. Berikut ini kita asumsikan bahwa input telah diubah ke bilangan bulat positif.

Dengan adanya petunjuk bahwa panjang input bisa mencapai 50 digit, maka kita tidak bisa menyimpan input dalam suatu variabel bilangan biasa. Longint hanya bisa menampung maksimum 10 digit bilangan bulat. `Comp` (dalam Turbo Pascal) atau `Int64` (dalam Free Pascal) hanya sanggup menampung 19 digit bilangan bulat. Mungkin ada yang beranggapan bahwa variabel-variabel *floating point* seperti `Real` (dalam Pascal) atau `Double` dapat menampung bilangan bulat tersebut. Hal itu memang benar, tapi yang disimpan hanya bisa sampai 23 digit pertama beserta tingkat eksponennya dalam basis 10 (sistem desimal). Bila pembagian dilakukan, maka hasilnya hanya akurat sampai digit ke 22. Dengan demikian, kita harus memikirkan suatu teknik penyimpanan dengan menggunakan tipe variabel dasar yang tersedia.

Adalah suatu yang alami bila kita berpikir untuk menggunakan array. Misalkan input kita simpan dalam array dari integer dengan panjang 50. Setiap digit dari input kita simpan dalam 1 elemen. Sekarang timbul masalah baru, yaitu bagaimana cara mendapatkan setiap digit dari input. Ada beberapa cara. Salah satunya adalah untuk membaca tiap baris sebagai sebuah string. Setelah itu, setiap karakter dari string tersebut dikonversi ke nilai digit menggunakan nilai ASCII dari karakter tersebut. Kemudian, panjang string itu menjadi panjang bilangan.

Setelah kita mendapatkan input tersebut dalam array, maka kita tinggal melakukan pembagian panjang dengan cara sesuai dengan yang telah diajarkan sewaktu kita masih duduk di sekolah dasar. Berikut ini kode semu (*pseudocode*) langkah-langkah untuk memecahkan soal ini. Sekedar catatan: “\” adalah pembagian yang hanya menghasilkan bilangan bulatnya saja.

```
Bil = jawab = array dari integer dengan panjang 50.
Pan Bil = Panjang dari Bil
Pan jawab = Panjang dari jawab = 0
Ubah input ke dalam array Bil
Sisa = 0

DARI I = 1 SAMPAI DENGAN Pan Bil LAKUKAN
 Sisa = Sisa x 10 + Bil[I]
 Pan jawab = Pan jawab + 1
 jawab[Pan jawab] = Sisa \ 13
```

Sebuah catatan kecil, untuk mencetak jawabannya, masih diperlukan ketelitian karena dalam ada kasus di mana jawabannya = 0 dan kasus di mana jawab[1] = 0 dan jawab[2] = 0.

Contoh 6.

### **Warisan** (Sumber: CEOI 1995)

Dua orang bersaudara, Pak Dingklik dan Pak Dengklek ingin membagi beberapa barang warisan yang didapat dari almarhum orang tuanya. Setiap barang yang ada, harus diberikan kepada Pak Dingklik dan Pak Dengklek. Hanya saja, masalahnya setiap barang warisan tersebut tidak dapat dibagi menjadi dua. Supaya adil, masing-masing warisan tersebut diberikan sebuah nilai dalam bentuk bilangan bulat yang positif. Misalkan A dan B melambangkan total nilai dari keseluruhan warisan yang diterima oleh Pak Dingklik dan Pak Dengklek. Tujuan yang hendak dicapai adalah meminimalkan selisih dan A dan B. Buat sebuah program untuk menghitung nilai A dan B.

#### **FORMAT MASUKAN**

Baris pertama dari file WARIS.IN terdiri dari N, jumlah dari warisan yang ada ( $1 \leq N \leq 100$ ). Baris berikutnya berisi N bilangan bulat positif, yang merupakan nilai dari setiap warisan. Tiap nilai  $\leq 200$ .

#### **FORMAT KELUARAN**

Tulis pada file WARIS.OUT nilai A dan B ( $A < B$ )

#### **CONTOH MASUKAN**

```
7
28 7 11 8 9 7 27
```

#### **CONTOH KELUARAN**

```
48 49
```

## PEMBAHASAN

(Oleh Windra Swastika, Mantan peserta seleksi TOKI)

Jika masalah ini diselesaikan secara brute force (dengan mencoba semua kemungkinan yang ada), maka akan mempunyai kompleksitas sebesar  $O(2^N)$ . Jika  $N=100$ , maka komputasi akan sangat memakan waktu yang besar. Untuk itu penyelesaian secara brute force adalah tidak disarankan untuk soal ini.

Pendekatan yang dilakukan adalah secara pemrograman dinamis (Dynamic Programming /DP). Konsep DP adalah menyelesaikan masalah dengan membagi menjadi sub masalah. Dari sub masalah yang terkecil (yang lebih mudah dicari nilai optimalnya), proses akan berlanjut untuk memecahkan masalah di atasnya (dengan memanfaatkan nilai optimal dari masalah sebelumnya). Nilai-nilai optimal tadi disimpan dalam sebuah tabel. DP menukar space dengan time!

Untuk soal ini, sebuah tabel yang mempunyai kombinasi penjumlahan untuk  $j=1, j=2, j=3 \dots j=N$ .

TABEL[j] = 0 jika  $j=0$

TABEL[j] = 1..N artinya warisan yang terakhir yang diberikan untuk mendapatkan subtotal j

TABEL[j] = 101 artinya subtotal j tidak bisa dihasilkan

Misalkan W adalah array untuk menyimpan nilai dari setiap warisan. Nilai dari subtotal j akan ditambahkan untuk mendapatkan subtotal baru ( $j+W[i]$ ) dengan syarat: TABEL[j]<i (nilai j adalah hasil dari subtotal yang pernah dicapai sebelumnya).

## SOLUSI

```
{
=====
-- WARI.S. PAS --
=====
Wi ndra Swasti ka
}

const
 finput='wari.s.in';
 foutput='wari.s.out';
 kosong=101;

var
 tabel: array [0..10000] of byte;
 warisan: array [1..100] of byte;
 f: text;
 i: byte;
```

```
j: word;
n: byte;
sum: word;

begin
 assign(f, finput);
 reset(f);
 readln(f, n);
 for i:=1 to n do
 begin
 read(f, warisan[i]);
 inc(sum, g[i]);
 end;
 close(f);

 fillchar(tabel, sizeof(tabel), kosong);
 tabel[0]:=0; {nilai awal untuk tabel[0]}
 for i:=1 to n do
 for j:=0 to sum shr 1-warisan[i] do
 if (tabel[j]<i) and (tabel[j+warisan[i]]=kosong) then
 tabel[j+warisan[i]]:=i;

 assign(f, foutput);
 rewrite(f);
 for i:=sum shr 1 downto 0 do {cukup mulai dari setengah sum}
 if tabel[i]<>kosong then
 begin
 writeln(i, ' ', sum-i);
 break;
 end;

 close(f);
end.
```

## Contoh 7.

### **Firetruck** (Sumber : ACM 1991 - Problem A)

Departemen pemadam kebakaran kota Arjosari bekerja sama dengan dinas transportasi lokal untuk mengolah peta yang menunjukkan status jalan-jalan pada kota Arjosari. Sialnya, jalan-jalan di kota itu pada hari tertentu harus ditutup karena adanya pawai mingguan. Akibatnya, para petugas pemadam kebakaran harus mencari rute jalan yang paling pendek jika hendak memadamkan api pada jalan tertentu di kota Arjosari.

Jika pada suatu saat, departemen pemadam kebakaran menerima laporan terjadinya kebakaran pada jalan tertentu, maka pihak departemen pemadam kebakaran segera meminta daftar jalan-jalan yang dapat dilalui ke dinas transportasi. Dinas transportasi mengirimkan semua jalan yang pada saat itu bisa dilalui. Buat sebuah program untuk mencari semua rute yang diawali dari kantor pusat departemen pemadam kebakaran menuju jalan yang dituju.

#### **FORMAT MASUKAN (fire.in)**

Baris pertama berisi bilangan bulat yang merupakan jalan terdekat dari pusat api yang harus dituju ( $2 \leq N < 21$ ).

Baris-baris berikutnya merupakan pasangan bilangan bulat kurang dari 21 (dipisahkan dengan spasi) yang menunjukkan jalan yang dapat dilalui. Baris terakhir pada file input adalah pasangan bilangan 0 0. Misalkan pasangan jalan 4 7 ada pada file input, artinya bahwa jalan 4 dan 7 dapat dilalui, begitu juga sebaliknya.

#### **FORMAT KELUARAN (fire.out)**

Semua kemungkinan rute yang dapat menuju ke N yang dimulai dari 1.

Setiap baris berisi urutan jalan yang harus ditempuh diawali dari 1 dan diakhiri di N. Agar perjalanan petugas pemadam kebakaran efisien, truk pemadam kebakaran tidak boleh menempuh jalan yang pernah dilalui lebih dari sekali.

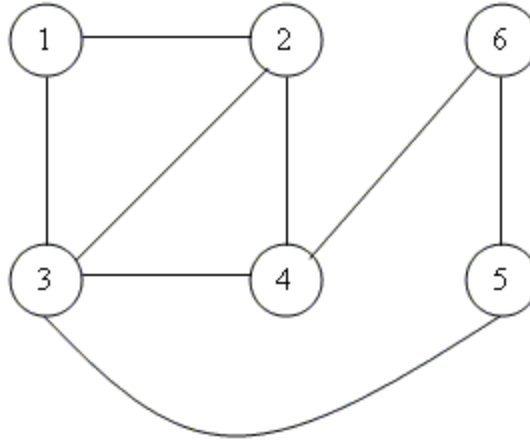
**CONTOH MASUKAN**

```
6
12
13
34
35
46
56
23
24
00
```

**CONTOH KELUARAN**

```
1 2 3 4 6
1 2 3 5 6
1 2 4 3 5 6
1 2 4 6
1 3 2 4 6
1 3 4 6
1 3 5 6
```

Soal ini termasuk soal-soal yang menggunakan dasar teori graf. Pada teori graf istilah dikenal verteks dan edge. Yang dimaksud dengan verteks adalah titik dan edge adalah garis. Sebuah graf dinotasikan dengan  $e=[u,v]$ , yang berarti bahwa edge  $e$  berawal pada verteks  $u$  dan berakhir pada verteks  $v$ . Pada Firetruck, terdapat maksimal 21 verteks. Sehingga contoh pada file input dapat dinotasikan  $e1=[1,2]$ , yang artinya edge  $e1$  berawal pada verteks 1 dan berakhir pada verteks 2. Dari file input, kita bisa mendapatkan sebuah graf seperti pada gambar berikut:



Dengan diagram seperti di atas, kita dapat membuat sebuah struktur data yang berisi jalur untuk masing-masing verteks. Dalam teori graf, jalur tersebut disebut dengan matriks jalur (*path matrix*).

```
const
 max_route=100
type
 tarrstreet=array[1..max_route, 1..max_route] of boolean
var
 arrstreet: tarrstreet;
```

Variabel `arrstreet` adalah sebuah matriks jalur (array dengan tipe boolean) yang menunjukkan adanya jalur dari verteks ke verteks. Dengan struktur data tersebut serta menggunakan teknik pencarian DFS (Depth First Search), maka dapat dengan mudah dibuat prosedur penelusuran dari verteks awal (1) menuju verteks tertentu.

```
const
 finput='fire.in';
 foutput='fire.out';
 max_route=100;
{
 =====
 -- FIRE.PAS --
 =====
 W i n d r a S w a s t i k a
}
```

```

type
 tstreet=1..21;
 troute=array[0..max_route] of tstreet;
 tarrstreet=array[1..max_route, 1..max_route] of boolean;
var
 fi, fo: text;
 map: tarrstreet;
 used: array[1..max_route] of boolean;
 destination: tstreet;
 i, j: byte;
 N: byte;
 route: troute;

procedure seekroute(depth: byte; start: tstreet);
var
 i: byte;
begin
 if start=destination then
 begin
 for i:=0 to depth-1 do
 write(fo, route[i], ' ');
 writeln(fo);
 end else
 for i:=1 to N do
 begin
 if map[start, i] and not used[i] then
 begin
 used[start]:=true;
 route[depth]:=i;
 seekroute(depth+1, i);
 used[start]:=false;
 end;
 end;
 end;
 end;
begin
 assign(fi, 'fire.in');
 reset(fi);
 readln(fi, destination);

 fillchar(map, sizeof(tarrstreet), false);
 fillchar(used, sizeof(used), false);
 fillchar(route, sizeof(troute), 0);
 N:=0; {number of street}

 repeat
 readln(fi, i, j);
 if (i>=0) and (j>=0) then
 begin
 map[i, j]:=true;
 map[j, i]:=true;
 inc(N);
 end;

```

```
until (i=0) and (j=0);
close(fi);

assign(fo, 'fire. out');
rewrite(fo);
route[0]:=1; {start from street #1}
used[1]:=true; {street #1 already used}
seekroute(1, 1);

close(fo);
end.
```



# Index

---

## A

algoritma · 2, 7, 49, 80  
Algoritma · 7  
append · 44, 45  
Array · 39, 40  
ASCII · 16, 75, 76, 87  
Assign · 44, 47, 64, 65  
Assignment · 11, 44

---

## B

bahasa assembly · 2  
Bahasa assembly · 2  
bahasa mesin · 2, 3, 4  
bahasa Pascal · 4, 5, 9, 10, 11, 16,  
18, 20, 57  
biner · 14, 19, 43  
Boolean · 13, 18, 19  
bug · 4

---

## C

Case · 24, 56  
case of · 25  
Character · 40  
**compile time** · 38

---

## D

debugger · 1, 4  
deklaratif · 1

---

## E

Editor · 4  
eksekusi · 1, 3, 4, 27, 38  
Eksekusi · 2  
**EKSEKUSI** · 1  
Ekspresi · 8, 15, 18, 21  
Elemen · 7  
Enum · 37  
**Enumerasi** · 37  
eof · 46, 77

---

## F

faktorial · 49, 50  
false · 18, 19, 28, 46, 77, 81, 82, 95  
File · 8, 43, 44, 45, 46, 73, 75, 79, 83  
FOR · 27  
FreePascal · 5, 13, 18, 22  
Fungsi · 18, 22, 31, 34, 46, 49, 62, 63  
fungsional · 1

---

## G

global · 9, 33

---

## H

hardware · 2, 4  
hardware) · 4  
himpunan · 38, 39, 54, 55, 59, 79

---

## I

IDE · 4  
Identifier · 10  
if - then · 23  
if - then - else · 23  
Indeks loop · 27  
input · 8, 13, 32, 44, 45, 46, 47, 59,  
74, 76, 77, 78, 83, 84, 86, 87, 92,  
93  
Input · 7, 8, 13  
Instruksi · 8, 23, 24  
Integer · 13, 14, 15, 19, 37, 53, 61,  
63, 66, 67, 68  
Integrated Development  
Environment · 4  
interpreter · 1, 2, 3, 4

Interpreter · 2, 3, 4  
 intersection · 39  
 Invarian Loop · 29  
 iteratif · 50

---

## K

Karakter · 13, 15, 16, 43  
 keyword · 9, 68  
 kompilator · 1, 2, 3, 4, 5, 9, 10, 18,  
 22, 28, 38, 40  
 Kompilator · 3, 4, 5  
 Konstanta · 8, 12  
 Konversi · 21, 22

---

## L

lokal · 33, 92  
 loop · 14, 27, 28, 29, 40, 41, 46, 50,  
 80

---

## M

mekanisme · 1  
**MEKANISME** · 1  
 MOD · 20  
 Moduler · 8

---

## N

NESTED LOOP · 28

---

## O

objek · 1  
 operand · 8, 15  
 operator · 8, 15, 19, 54, 55, 58, 68,  
 69  
 ord · 16, 40  
 output · 8, 13, 45, 46, 57, 65, 66, 74,  
 76, 77, 78, 84  
 Output · 7, 8, 13, 59, 60

---

## P

paradigma · 1, 2, 7  
 Paradigma · 1, 2  
**PARADIGMA** · 1  
 parameter · 32, 33, 34, 45, 46, 52,  
 63, 67, 68  
 Parameter · 32, 33  
 pemrograman · 1, 5, 7, 8, 11, 12, 13,  
 15, 31, 34, 38, 90, 100  
**PEMROGRAMAN** · 1, 7  
 precedence · 15  
 program · 1, 2, 3, 4, 5, 8, 9, 10, 11,  
 12, 17, 24, 27, 29, 31, 33, 34, 38,  
 43, 44, 45, 47, 55, 56, 57, 58, 59,  
 60, 61, 62, 63, 64, 65, 66, 67, 68,  
 69, 70, 73, 74, 76, 77, 78, 81, 83,  
 84, 89, 92  
**PROGRAM** · 1, 9, 66  
 programmer · 2, 4  
 prosedur · 10, 13, 22, 31, 32, 33, 34,  
 44, 45, 49, 52, 54, 94  
 prosedural · 1, 2, 7, 8, 12, 31, 38  
 proses · 4, 8, 16, 19, 21, 44, 90  
 Proses · 4, 7, 8, 44

---

## R

read · 13, 14, 39, 44, 46, 77, 91  
 readln · 13, 14, 18, 37, 38, 39, 44, 45,  
 46, 47, 59, 74, 76, 77, 78, 81, 95  
 real · 11, 14, 19, 20, 21, 22, 37, 39,  
 52, 53, 68  
 Real · 13, 19, 20, 62, 63, 68, 87  
 reference · 34  
 rekurens · 49, 50  
 Rekurens · 8  
**REKURENS** · 49  
 rekursif · 49, 50  
 REPEAT - UNTIL · 28  
 representasi · 1, 12, 14, 17, 19, 21  
 reset · 44, 45, 47, 74, 77, 78, 81, 91,  
 95  
 rewrite · 44, 45, 47, 74, 77, 78, 82,  
 91, 96  
**run time** · 38

---

## S

Scope · 33  
 seksadesimal · 38  
 sekuensial · 2, 8  
 SET · 38  
 sintaks · 2, 40  
 software · 4, 76, 100  
**STDIN** · 46  
**STDOUT** · 46  
 String · 13, 17, 40, 53, 61, 63, 69, 70,  
 75, 78  
 Struktur Data · 8  
 subprogram · 31  
 Subtipe · 37, 38

---

**T**

Tabel · 39, 40, 84, 85

teks · 4, 43, 44, 45

Tipe · 7, 12, 13, 14, 16, 18, 19, 20,  
21, 37, 38, 39, 52, 53

TOKI · 31, 43, 46, 47, 60, 61, 62, 80,  
90, 100

true · 18, 19, 28, 46, 77, 81, 82, 95,  
96

---

**U**

unary · 15

underscore · 10

**UNICODE** · 16, 17

union · 39, 54

---

**V**

valid · 10, 38, 53, 57, 67

variabel · 10, 11, 12, 13, 14, 15, 17,  
20, 33, 34, 38, 39, 43, 44, 53, 54,  
63, 64, 67, 68, 70, 76, 77, 84, 87

Variabel · 8, 11, 13, 20, 33, 35, 54,  
68, 70, 94

---

**W**

**WHILE - DO** · 27

write · 13, 32, 39, 45, 46, 58, 59, 69,  
74, 95

writeln · 10, 12, 13, 16, 17, 18, 20,  
21, 23, 24, 27, 29, 31, 32, 33, 37,  
39, 40, 45, 46, 47, 53, 56, 58, 59,  
60, 61, 63, 69, 74, 77, 78, 82, 95



## BIODATA SINGKAT PENULIS



***Yohanes Nugoro***, Sarjana Teknik Informatika lulus tahun 2002 dari Departemen Informatika Institut Teknologi Bandung saat ini bekerja sebagai teaching assistant di Departemen Teknik Informatika ITB dan merupakan salah satu pembina di TOKI Biro ITB. Saat ini juga sedang menyelesaikan studi lanjutan S2-nya di institusi yang sama.

Selain menguasai hampir semua bahasa pemrograman, penulis adalah salah satu Pengajar/Pembina di Tim Olimpiade Komputer Indonesia, yang handal. Penulis juga aktif pada kegiatan Opensource di Indonesia dan banyak terlibat dalam proyek-proyek pembangunan software., serta aktif menghasilkan berbagai macam tulisan/artikel tentang teknologi informasi.