TURBO PASCAL

Sebuah Pengantar pada Komputer Pemrograman
(TP 7.0 dan TPW 1.5)

Oleh : *Pramudiyanto, S.Pd.T.*

WHAT'S TURBO PASCAL?

Sebuah software bahasa pemrograman yang sangat interaktif. Merupakan salah satu kompiler yang terkenal untuk pemrograman. Program yang ditulis dengan software ini memiliki ciri yang terstruktur, sehingga mudah dipahami maupun dikembangkan untuk pemrogram. Dikembangkan oleh *Borland International*, bahasa Pascal termasuk *high level language*, yang berorientasi pada bahasa manusia, dengan menggunakan bahasa Inggris sebagai bahasa dasarnya.

Beberapa bahasa pemrograman yang mirip seperti Turbo Pascal antara lain :

- BASIC (Q-Basic, Turbo Basic, GW Basic)
- FORTRAN
- COBOL
- ASSEMBLER
- PERL
- dll

Kemudahan dalam memahami dan mempelajari bahasa Pascal ditunjang oleh bentuk program Pascal yang terstruktur. Sebagai bahasa yang terstruktur, program Pascal tersusun atas sejumlah blok. Blok-blok yang kecil selanjutnya dapat dipakai untuk membuat blok-blok yang lebih besar, dan secara keseluruhan membentuk program kerja. Hal ini memberikan kemudahan bagi pemrogram dalam membuat, mengembangkan, dan memahami program tersebut. Suatu permasalahan dapat dipecah-pecah menjadi bagian-bagian yang kecil sehingga dapat dengan mudah dikodekan. Keuntungan lain, logika program menjadi lebih mudah dipelajari/dimengerti. Kesalahan-kesalahan yang terjadi di dalam program akan mudah ditelusuri. Disamping itu, program menjadi lebih mudah dimodifikasi tanpa khawatir menimbulkan efek sampingan terhadap bagian lain dari program.

FASILITAS TURBO PASCAL : KOMPILER SEBAGAI TRANSLATOR

Program yang berorientasi pada *high level language* seperti Pascal, sebenarnya tidak langsung dimengerti oleh komputer, sebab komputer hanya mengerti satu bahasa saja, yaitu bahasa mesin. Agar program yang telah tersusun dapat dijalankan di komputer, maka program yang telah tersusun harus diterjemahkan terlebih dahulu ke dalam bahasa mesin oleh suatu piranti yang dinamakan **translator**. Sedangkan translator dapat berupa *Interpreter* maupun *Kompiler*.

Interpreter menterjemahkan instruksi selama eksekusi program. Jika dikehendaki untuk menjalankan program, mula-mula program sumber (*source program* atau program yang ditulis dalam *high level language*) diterjemahkan terlebih dahulu ke dalam bentuk kode mesin per instruksi. Setelah instruksi tersebut dipahami oleh komputer dan dijalankan, translator kembali mengulang proses serupa untuk instruksi berikutnya. Dengan cara seperti itu, suatu instruksi akan dijalankan dengan sangat lambat.

Berbeda dengan interpreter, kompiler menterjemahkan instruksi secara keseluruhan terlebih dahulu ke dalam kode mesin sebelum program dapat dijalankan. Setelah penterjemah (dalam hal ini dinamakan **kompilasi**), kompiler tidak diperlukan lagi, sebab hasil penterjemahan bersifat *executable*, artinya dapat dieksekusi langsung pada sistem operasi. Melalui langkah ini, program akan dijalankan dengan relatif lebih cepat, sebab tidak terdapat proses penterjemahan lagi.

Fasilitas kompiler pada Turbo Pascal tidak hanya sekedar kompiler, tetapi juga mengandung editor teks. Bahkan eksekusi program bisa dilaksanakan dari Turbo Pascal, tanpa harus keluar ke sistem operasi. Fasilitas yang lain, program sumber yang ditulis tidak harus disimpan terlebih dahulu apabila ingin dikompilasi. Apabila program dirasa sudah cukup memadai, barulah program sumber tersebut disimpan ke dalam bentuk file. Cara ini tentu saja menghemat waktu. Selain itu, Turbo Pascal memungkinkan hasil kompilasi dapat dipilih dan ditempatkan ke memori (RAM) atau ke media penyimpanan permanen (disk).

Kompilasi ke memori (RAM) memiliki kelebihan : proses kompilasi sangat cepat. Namun karena tidak dapat disimpan, tentu saja eksekusi program hanya dapat dijalankan pada Turbo Pascal. Oleh karena itu, biasanya kompilasi ke memori (RAM) hanya dilakukan selama pengujian program saja, setelah itu, baru dikompilasi ke dalam media penyimpanan permanen (disk), sehingga dapat dieksekusi langsung dari sistem operasi.

STRUKTUR PROGRAM DALAM TURBO PASCAL

Secara umum, struktur program dalam Turbo Pascal dapat digambarkan sebagai berikut :

```
Program BAGAN_PROGRAM; ← bagian dari kepala program
Uses ..  ← menyatakan unit yang dilibatkan dalam program
{ semua yang berada dalam tanda kurung ini dianggap sebagai komentar dan tidak akan
diproses }
(* begitu pula dengan seluruh informasi yang ada dalam tanda ini *)

Label ..
Const ..
Type ..
Var ..          } bagian deklarasi
Procedure ..
Function ..

Begin { awal dari program }
.
.
{ statement-statement dari BAGAN_PROGRAM } } bagian pernyataan
.
.
end. { akhir dari sebuah program }
```

Secara umum, struktur program Turbo Pascal dibagi menjadi 3 bagian, yaitu : kepala program, definisi dan deklarasi, dan bagian pernyataan. Program biasanya diawali dengan nama program (diawali dengan kata baku/*reserved word* **PROGRAM**), diikuti dengan judul program dan diakhiri dengan tanda titik koma (;). Setelah bagian kepala program dapat mengandung klausa **USES**. Klausa ini jika disertakan menunjukkan adanya unit yang dilibatkan di dalam program. Pengertian unit sendiri sebenarnya suatu modul dalam file yang berisi pustaka (*library*), biasanya mengandung fasilitas berupa prosedur atau fungsi, yang dapat digunakan dalam program tanpa harus menuliskannya kembali dalam program.

Setelah klausa **uses**, sebuah program dapat diikuti dengan deklarasi statemen label (diawali dengan kata baku **LABEL**), deklarasi konstanta (diawali dengan kata baku **CONST**), deklarasi tipe data (diawali dengan kata baku **TYPE**), dan deklarasi peubah dari program utama (diawali dengan kata baku **VAR**), prosedur-prosedur, fungsi-fungsi, dan akhirnya bagian program utamanya sendiri (bagian pernyataan).

Bagian deklarasi prosedur dan fungsi sama dengan deklarasi program utama. Artinya baik prosedur maupun fungsi dapat memiliki deklarasi label, konstanta dan lain-lain, yang berbeda dengan deklarasi yang diperuntukkan bagi program utama.

TOKEN dan KONSTANTA

1. Simbol Khusus dan Kata Baku

Token adalah unit terkecil dari teks dalam program Pascal yang memiliki arti khusus, dan dikelompokkan menjadi simbol, pengenal (*identifier*), label, bilangan dan konstanta untai (*string constant*). Dalam Turbo Pascal, token dibentuk dengan menggunakan sejumlah karakter yang merupakan sub himpunan dari himpunan karakter ASCII. Karakter yang digunakan adalah

- Huruf : a sampai dengan z, dan A sampai dengan Z
- Digit : angka Arab 0 sampai dengan 9
- Digit heksadesimal : angka Arab 0 sampai dengan 9, huruf A sampai dengan F, dan huruf a sampai dengan f.
- Spasi (*blank*) : karakter spasi (ASCII 32) dan semua karakter kendali (ASCII 0 sampai dengan 31), termasuk karakter akhir baris (ASCII 13).

Simbol-simbol khusus dan kata baku adalah karakter-karakter yang memiliki satu atau lebih arti yang lengkap. Berikut ini adalah karakter-karakter khusus :

+ - * / = < > [ ] . , ( ) : ; ^ @ { } $ #

Pasangan karakter-karakter khusus berikut ini juga merupakan simbol-simbol khusus :

<= >= := .. (* *) (. .)

2. Pengenal

Pengenal digunakan untuk menunjukkan konstanta, tipe, peubah, prosedur, fungsi, unit, program, dan medan-medan dalam rekaman. Panjang pengenal dapat terdiri dari beberapa karakter, namun yang digunakan adalan 63 karakter yang pertama. Pengenal harus diawali dengan huruf, diikuti huruf lain, digit, atau garis bawah dan tidak boleh diberi spasi. Sebagai contoh pengenal `Nama_Siswa`, `Program_Latihan`, `A123`, adalah contoh pengenal yang benar, sedangkan `Nama Siswa`, `1ABC`, adalah contoh pengenal yang salah.

3. Label

Label adalah deretan digit antara 0 sampai dengan 9999. Digit 0 pertama tidak diperhatikan. Label ini digunakan oleh statemen `goto` untuk melompatkan suatu proses eksekusi ke suatu statemen tertentu. Dalam Turbo Pascal, pengenal juga dapat berfungsi sebagai label.

4. Bilangan

Konstanta bilangan real atau bulat biasanya dinyatakan dalam sistem bilangan desimal. Konstanta bilangan bulat juga sering dituliskan menggunakan sistem bilangan heksadesimal yang diawali dengan tanda $. Konstanta bilangan real juga dapat ditulis menggunakan notasi eksponensial. Sebagai contoh, bilangan `7E-2` artinya adalah $7x10^{-2}$, bilangan `12.9E+4` atau `12.9E4` mempunyai arti yang identik dengan $12.9x10^{4}$. Bilangan yang memiliki titik desimal atau yang ditulis dengan menggunakan eksponensial selalu berupa bilangan real. Bilangan real dapat ditulis dengan notasi *fixed point*, misalnya `12.5`, atau menggunakan notasi *floating point*, misalnya `1.25e1`.

5. Untai Karakter

Untai karakter adalah deretan dari sejumlah karakter yang terdapat dalam tabel ASCII, yang harus ditulis diantara tanda kutip '. Untai karakter yang tidak memiliki apa-apa di antara tanda kutip tersebut disebut karakter kosong (*null string*). Panjang untai karakter dinyatakan sebagai banyaknya karakter yang ditulis dalam tanda kutip. Dalam Turbo Pascal, karakter-karakter kendali juga dapat dinyatakan dalam untai karakter. Karakter # diikuti dengan bilangan antara 0 sampai dengan 255 menyatakan karakter ASCII ssuai dengan bilangan tersebut. Dalam hal ini tidak boleh ada pemisah antara tanda # dengan bilangannya.

6. Konstanta

Deklarasi konstanta menunjukkan nilai yang tetap dari suatu pengenal dan berlaku pada blok dimana deklarasi tersebut dinyatakan. Bentuk umum dari deklarasi konstanta adalah :

```
Const pengenal = nilai;
```

Dengan *pengenal* : nama pengenal; dan *nilai* : nilai konstanta.

Dalam Turbo Pascal, deklarasi konstanta juga boleh berisi ungkapan. Konstanta yang demikian disebut dengan ungkapan konstanta (*constant expression*). Contohnya :

```
Const    awal = 0;
         Akhir = 100;
         Rata = (akhir-awal) div 2;
         Huruf = ['A'..'Z','a'..'z'];
         Angka = ['0'..'9'];
         Huruf_Angka = Huruf + Angka;
```

Dalam ungkapan konstanta juga diperbolehkan menggunakan beberapa fungsi standar. Fungsi standar yang dimaksud adalah `abs`, `chr`, `hi`, `length`, `lo`, `odd`, `pred`, `ptr`, `round`, `sizeOf`, `succ`, `swap`, dan `trunc`.

7. Baris Komentar

Baris komentar adalah suatu kalimat yang biasanya digunakan untuk menjelaskan antara lain kegunaan program, batasan-batasan, dan lain sebagainya. Baris komentar ini bersifat sebagai *unexecutable statement*. Cara penulisan baris komentar adalah sebagai berikut :

{ baris komentar } atau **(* baris komentar *)**

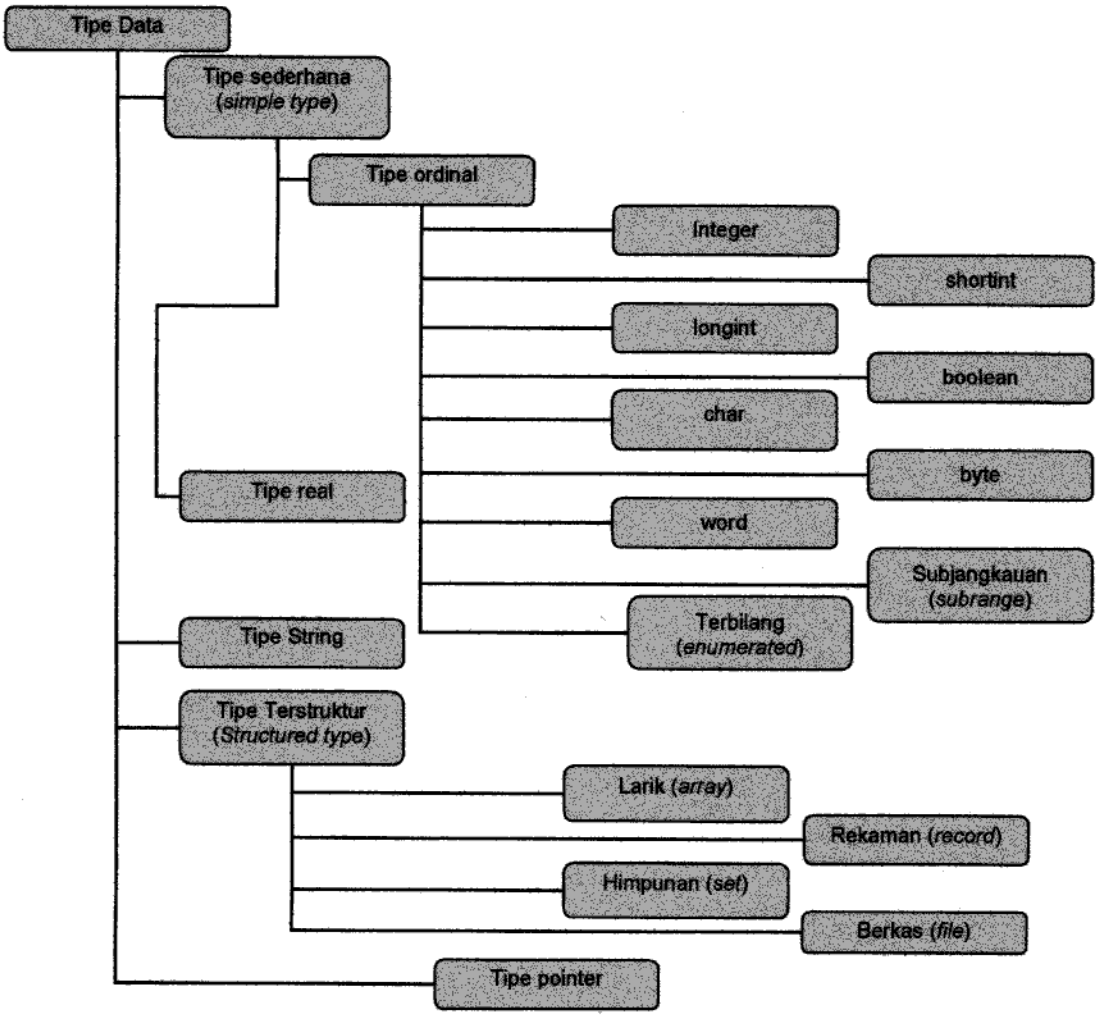
TIPE DATA

Dalam Pascal, semua peubah yang akan digunakan harus sudah ditentukan tipe datanya. Dengan menentukan tipe data suatu peubah, sekaligus menentukan batasan nilai peubah tersebut dan jenis operasi yang dapat dilaksanakan atas peubah tersebut. Bentuk umum dari deklarasi tipe data adalah :

```
Type pengenal = tipe;
```

Dengan *pengenal* : nama pengenal yang menyatakan tipe data; *tipe* : tipe data yang berlaku dalam Turbo Pascal.

Secara lengkap, tipe data dalam Turbo Pascal dapat digambarkan sebagai berikut :



1. Tipe Sederhana

Tipe sederhana juga disebut dengan tipe data skalar, yang dapat diartikan bahwa dalam sebuah peubah hanya dimungkinkan untuk menyimpan sebuah nilai data.

a. Tipe Ordinal

1) Tipe Integer

Tipe integer adalah tipe data yang nilainya tidak memiliki nilai desimal. Dalam Pascal hanya digit yang dapat muncul sebagai integer, dengan demikian tidak ada karakter lain (misal koma) yang diperbolehkan. Tanpa plus atau minus dapat mendahului bilangan tersebut. Terdapat 5 buah tipe data yang termasuk dalam kelompok tipe data integer, yaitu `shortint`, `integer`, `longint`, `byte`, dan `word`. Batas nilai masing-masing tipe di atas diberikan sebagai berikut :

Tipe	Batas nilai	Ukuran dalam byte
Byte	0 .. 255	1
Shortint	- 128 .. 127	1
Integer	-32768 .. 32767	2
Word	0 .. 65535	2
Longint	-2147483648 .. 2147483647	4

Seperti dikatakan di atas, tipe data juga menentukan macam operasi yang dapat dilaksanakan. Operator-operator yang dapat dikerjakan dengan tipe data integer adalah sebagai berikut :

Operator	Kegunaan
+	Penjumlahan
-	Pengurangan

Operator	Kegunaan
*	Perkalian
div	Pembagian
mod	Sisa pembagian

2) Tipe Boolean

Data tipe boolean hanya memiliki dua nilai, yaitu **benar (TRUE)** dan **salah (FALSE)**. Dengan menggunakan operator `and`, `or`, atau `not` dapat dibentuk ungkapan boolean yang lebih rumit. Nilai boolean sangat penting untuk pengambilan keputusan dalam suatu program.

3) Tipe Char

Dalam Turbo Pascal, kata char digunakan untuk mendefinisikan tipe data yang nilainya merupakan himpunan karakater yang dikenal komputer. Dalam program konstanta bertipe char ditulis di antara tanda petik, misalnya

'A' 'B' ',' '4'

Hal yang perlu diingat bahwa data bertipe char hanya terdiri atas 1 karakter.

4) Tipe Terbilang

Disebut tipe terbilang karena semua nilai disebut satu per satu. Sebagai contoh :

```
Type Toko = (BARU,RAMAI,SUKSES,RAPI,GADJAH)
```

Perlu diperhatikan bahwa dalam tipe terbilang, semua data harus diletakkan di antara tanda kurung. Urutan dalam tanda terbilang perlu diperhatikan karena akan mempengaruhi nilai fungsi pred dan succ. Dengan menggunakan deklarasi seperti di atas, dapat dilihat bahwa

`Pred(RAMAI)` adalah `BARU`
`Succ(RAMAI)` adalah `SUKSES`

Data baru dengan tipe Toko seperti di atas dapat digunakan untuk menjelaskan peubah lain yang juga bertipe Toko. Hal ini memungkinkan kita untuk memakai nama-nama toko dalam program.

5) Tipe Subjangkauan

Tidak jarang terjadi batas nilai yang mungkin untuk suatu peubah merupakan bagian (subjangkauan) dari tipe data yang telah didefinisikan. Sebagai contoh, jangkauan yang mungkin dari nilai ujian adalah 0 sampai dengan 100, dan ini hanya merpuakan bagian jangkauan dari data bertipe integer. Data bertipe subjangkauan dapat didefinisikan pada tipe ordinal yang sebelumnya telah didefinisikan terlebih dahulu. Deklarasi tipe subjangkauan mempunyai bentuk umum :

```
Type pengenal = konstanta1 .. konstanta2;
```

Dengan *pengenal* : nama tipe data yang akan dideklarasikan
Konstanta1 : batas bawah nilai data
Konstanta2 : batas atas nilai data

Sebagai contoh, tipe subjangkauan Nilai dapat didefinisikan sebagai :

```
Type Nilai = 0 .. 100;
```

b. Tipe Real

Konstanta bertipe real adalah bilangan yang berisi titik desimal. Dalam Pascal, paling sedikit harus ada satu digit sebelum dan sesudah titik desimal. Tidak boleh ada koma dan nilainya bisa positif maupun negatif. Bilangan real juga dapat dinyatakan dalam bentuk eksponensial. Dalam pemakaiannya pangkat dari bilangan dasar 10 (yang digunakan

untuk menunjukkan eksponensial) dinyatakan dengan notasi E. Jika dinyatakan dalam notasi eksponensial, maka bilangan 0.00017543 adalah 1.7543E-4.

Dalam Turbo Pascal terdapat lima macam tipe real, yaitu real, single, double, extended, dan comp. Tipe single, double, extended, dan comp hanya dapat digunakan apabila komputer dilengkapi dengan *numeric coprocessor* 80x87, meskipun demikian, untuk generasi saat ini, biasanya komputer telah dilengkapi dengan *numeric coprocessor* 80x87. Tipe-tipe real dengan batas-batas nilainya diberikan dalam bentuk tabel sebagai berikut :

Tipe	Batas nilai	Angka Signifikan	Ukuran dalam byte
Real	2.9x10E-39 .. 1.7x10E38	11 – 12	6
Single	1.5x10E-45 .. 3.4x10E38	7 – 8	4
Double	5.0x10E-324 .. 1.7x10E308	15 – 16	8
Extended	1.9x10E-4951 .. 1.1x10E4932	19 – 20	10
Comp	-2E63 + 1 .. 2E63 + 1	19 – 20	8

Operasi yang dapat dilakukan pada tipe bilangan real sebagai berikut :

Operator	Kegunaan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian

Bilangan-bilangan real banyak digunakan dalam perhitungan-perhitungan matematika, sains, rekayasa, dimana derajat ketelitian yang sangat tinggi sangat diperhatikan. Beberapa kesalahan mungkin akan terjadi sehubungan dengan pemakaian bilangan real karena bilangan real biasanya dinyatakan dalam *scientific numbering* yang memiliki cacah digit yang tetap. Beberapa bilangan memerlukan cacah digit yang tak terhingga. Sebagai contoh, pecahan 1/3 yang dinyatakan dalam bilangan real, akan memiliki bentuk 0.333333333..

Komputer biasanya memiliki perangkat keras untuk operasi penjumlahan, pengurangan, perkalian dan pembagian pada bilangan real. Karena penyajian bilangan integer berbeda untuk melakukan operasi atas dua tipe data ini. Untuk membangkitkan kode untuk melakukan operasi terhadap data bertipe real, Turbo Pascal menggunakan bantuan perangkat lunak (*software floating point*) dan perangkat keras (*hardware floating point*), yang dapat dipilih menggunakan petunjuk kompiler $N. Untuk menggunakan perangkat lunak, digunakan petunjuk kompiler -$N, yangmerupakan standarnya. Sedangkan untuk perangkat keras digunakan +$N. Dalam hal ini, kelima tipe data yang disebutkan di atas dapat digunakan dengan catatan komputer telah dilengkapi dengan *numeric coprocessor* 80x87.

2. Tipe String

Data bertipe string adalah data yang berisi sederetan karakter yang banyaknya karakter dapat berubah-ubah sesuai kebutuhan, yaitu dari 1 sampai dengan 255 karakter. Tipe string yang tidak dinyatakan panjang karakternya dianggap memiliki 255 karakter. Bentuk umum dari deklarasi tipe string adalah :

```
Type pengenal = string <[panjang]>;
```

dengan *pengenal* : nama tipe data.
Panjang : bilangan bulat yang menunjukkan banyaknya karakter.

Apabila parameter *panjang* tidak ditulis, panjang karakter dianggap 255 karakter. Urutan dari dua buah string sembarang ditentukan berdasrkan posisi setiap karakternya. Karakter-karakter dalam string dapat dimasup seperti halnya komponen larik.

3. Tipe Terstruktur

Dalam tipe terstruktur, setiap peubah dapat menyimpan lebih dari sebuah nilai data. Masing-masing nilai data tersebut disebut dengan komponen. Tipe terstruktur karakteristiknya ditentukan berdasarkan penstrukturan dan tipe masing-masing komponen. Jika komponennya juga bertipe tersetruktur, tipe terstruktur yang dihasilkan memiliki lebih dari sebuah tingkat penstrukturan. Ukuran tipe terstruktur dalam Turbo Pascal maksimum 65520 byte. Pada tipe terstruktur terdapat 4 buah tipe data terstruktur, yaitu larik, rekaman, himpunan, dan berkas.

a. Tipe Larik

Larik (*array*) adalah tipe struktur yang memiliki komponen dalam jumlah yang tetap dan setiap komponen memiliki tipe data yang sama. Posisi masing-masing komponen dalam larik dinyatakan sebagai nomor indek. Bentuk umum dari deklarasi tipe larik adalah sebagai berikut :

```
Type pengenal = array[tipe_indek] of tipe;
```

dengan *pengenal* : nama tipe data
Tipe_indek : tipe data untuk nomor indek
Tipe : tipe data komponen

Parameter tipe_indek menentukan banyaknya komponen larik tersebut. Parameter ini boleh berupa sembarang tipe ordinal kecuali longint dan subjangkauan dari longint. Sebagai contoh :

```
Type vek = array [1..100] of integer;
```

Menunjukkan bahwa *vek* adalah tipe data yang berupa larik yang komponennya bertipe integer dan banyaknya 100 buah. Deklarasi yang demikian dinamakan deklarasi larik dimensi satu (disebut pula vektor). Apabila tipe komponen juga berupa sebuah larik lain, akan diperoleh larik dimensi banyak. Sebagai contoh deklarasi :

```
Type Tbl = array [1.100] of array [1..5] of real;
```

Menunjukkan bahwa *Tbl* adalah vektor yang terdiri dari 100 komponen, dengan tipe komponennya adalah sebuah vektor lain yang memiliki 5 buah komponen dan bertipe real. Bentuk seperti ini dinamakan larik dimensi dua (disebut pula sebagai tabel atau matrix).

Contoh lain misalnya :

```
Type Tbl = array [boolean,1..100,1..5] of char;
```

Disebut sebagai deklarasi larik dimensi 3.

b. Tipe Rekaman

Seperti halnya dengan tipe larik, rekaman (*record*) adalah kumpulan data. Perbedaan antara larik dengan rekaman adalah bahwa dalam larik semua elemennya harus bertipe sama, tetapi dalam rekaman setiap tipe dapat memiliki tipe data yang berbeda satu sama lain. Bentuk umum dari deklarasi rekaman sebagai berikut :

```
Type pengenal = record
                Medan_1     : tipe_1;
                Medan_2     : tipe_2;
                .
                .
                .
                medan_n     : tipe_n;
                    end;
```

dengan *pengenal* : pengenal yang menunjukkan tipe data yang akan dideklarasikan.
medan_1, … ,medan_n : nama medan yang akan digunakan.
Tipe_1, …, tipe_n : sembarang tipe data yang telah dideklarasikan sebelumnya.

Adakalanya diperlukan suatu bentuk rekaman yang salah satu medannya bervariasi tergantung dari kebutuhan. Rekaman yang demikian disebut dengan rekaman bebas (*variant record*). Bentuk umum rekaman bebas adalah :

```
Type pengenal = record
                {* bagian tetap *}
                medan_1      : tipe_1;
                medan_2      : tipe_2;
                .
                .
                .
                medan_n      : tipe_n;

                {* bagian bebas *}
                case tag     : tipe_tag of
                      label_1     :     (medan : tipe;
                                         .
                                         .
                                         .
                                         medan : tipe);
                      label_2 :    (medan : tipe;
                                         .
                                         .
                                         .
                                         medan : tipe);
                      label_3 :    (medan : tipe;
                                         .
                                         .
                                         .
                                         medan : tipe);
                end;
```

dengan

medan, medan_1, medan_2, …	: nama medan rekaman.
Tipe, tipe_2, tipe_3, …	: tipe data medan
Tag	: pengenal untuk pemilihan kasus
Tipe_tag	: tipe data dari pengenal untuk pemilihan kasus.
`Label_1, label_2, label_3,…`	: nama label yang menunjukkan kasus yang dipilih.

Dari bentuk umum di atas, dapat dilihat bahwa rekaman bebas terbagi menjadi dua bagian, yaitu bagian tetap dan bagian bebas, yaitu bagian yang akan dipilih sesuai dengan kasus yang dihadapi. Medan dalam bagian bebas sering disebut dengan *tag field*.

c. Tipe Himpunan

Himpunan adalah kumpulan objek yang memiliki tipe data yang sama dan urutan penulisannya tidak diperhatikan. Setiap objek di dalam suatu himpunan disebut dengan anggota atau elemen himpunan. Jika dibandingkan dengan larik yang dapat dioperasikan berdasar elemen-elemennya, maka himpunan selalu dioperaskan secara keseluruhan sebagai satu kesatuan. Dalam larik, urutan elemen sangat diperhatikan, tetapi untuk operasi-operasi terhadap himpunan urutan penulisan anggota sama sekali tidak diperhatikan.

Dalam Pascal, anggota himpunan diletakkan di dalam tanda kurung kotak, [ ]. Juga dimungkinkan adanya himpunan yang tidak memiliki anggota, yang dinamakan himpunan kosong dan ditulis dengan menggunakan notasi []. Bentuk umum dari deklarasi himpunan sebagai berikut :

```
Type pengenal = set of tipe_data;
```

Himpunan juga dapat langsung dideklarasikan dalam bagian deklarasi peubah :

```
Var pengenal : set of tipe_data;
```

dengan `pengenal` : nama peubah atau pengenal yang akan dinyatakan sebagai tipe himpunan

`Tipe_data` : tipe data dari anggota himpunan, harus bertipe ordinal.

Dua buah himpunan atau lebih dapat saling dioperasikan satu sama lain. Dalam Pascal dikenal tiga macam operasi biner terhadap himpunan, yaitu (ditulis secara hierarki operasi) :

- Interseksi, dengan operator *
- Union, dengan operasi +
- Selisih, dengan operator -

Selain operasi biner, dua buah himpunan juga dapat dioperasikan secara rasional. Operator rasional yang digunakan dalam data bertipe himpunan sebagai berikut :

Operator	Digunakan untuk
=	Test untuk kesamaan dua himpunan.
<>	Test untuk ketidaksamaan dua himpunan.
>=	Test apakah semua anggota dari operand kedua terdapat dalam operand pertama (superset).
<=	Test apakah semua anggota operand pertama terdapat dalam operand kedua (subset).
in	Test keanggotaan suatu himpunan. Operand pertama dapat berupa sembarang ungkapan dengan syarat bahwa tipe datanya harus sama dengan tipe operand kedua.

d. Tipe Berkas

Berkas (*file*) adalah kumpulan sejumlah komponen yang bertipe data sama, yang jumlahnya tidak tertentu, dan biasanya tersimpan dalam suatu media penyimpanan luar. Jumlah komponen dalam berkas dapat ditambah jika diperlukan. Pengertian berkas analog dengan simpanan arsip dalam lemari arsip yang setiap kali dapat ditambah, diambil untuk dibaca atau juga dihapus karena tidak digunakan lagi.

Di dalam Pascal, berkas menyediakan data yang nantinya akan digunakan oleh suatu program. Satu aspek yang penting dari berkas adalah bahwa data yang ada di dalam berkas dapat digunakan oleh sembarang program yang tipe datanya disesuaikan dengan kebutuhan. Hal ini semakin mempertinggi derajat keluwesan suatu berkas. Bentuk umum dari deklarasi berkas adalah :

```
Type pengenal = file of pengenal1;
```

dengan *pengenal* : pengenal yang akan dinyatakan sebagai tipe data berkas

Pengenal1 : tipe data komponen berkas.

Parameter pengenal1 yang menunjukkan tipe data komponen berkas yang disebut dengan tipe dasar (*base type*) dari berkas, dan sembarang tipe data dapat digunakan sebagai tipe dasar komponen berkas. Biasanya tipe dasar berkas berupa rekaman.

Jenis kedua dari berkas dalam Pascal adalah berkas teks (*text file*) yaitu berkas yang berisi deretan karakter. Berbeda dengan jenis berkas yang telah dijelaskan sebelumnya, komponen-komponen pada berkas teks membentuk suatu baris dan setiap baris diakhiri dengan tanda akhri baris (*end-of-line, carriage return,* atau *line-feed*). Setiap berkas teks selalu diakhiri dengan tanda akhir baris berkas (*end-of-file*). Pendeklarasian berkas teks hanya ditunjukkan dengan kata baku text,yang ditempatkan dalam bagian deklarasi type, yaitu :

```
Type pengenal = text;
```

Dengan *pengenal* adalah nama yang mewakili berkas teks.

4. Tipe Pointer

Semua tipe data yang telah dijelaskan sebelumnya, apabila digunakan untuk mendeklarasikan suatu peubah, maka sifat peubah tersebut adalah peubah yang statis. Pascal dilengkapi dengan fasilitas yang memungkinkan pemakai untuk menggunakan peubah yang bersifat dinamis, yang disebut dengan pointer. Namun di sini tidak akan dijelaskan bagamana bentuk dan penggunaan dari tipe pointer.

PEUBAH

Peubah sebenarnya mewakili suatu nilai data tertentu yang akan dioperasikan dalam suatu program. Seperti dijelaskan sebaelumnya setiap peubah harus dinyatakan tipe datanya. Bentuk umum deklarasi peubah sebagai berikut :

```
Var pengenal : tipe_data;
```

dengan *pengenal* : nama peubah yang akan dideklarasikan

Tipe_data : tipe data yang akan digunakan.

Nama peubah sebaiknya dipilih agar mudah diingat dan mempermudah pengecekan program apabila terjadi kesalahan.

KONSTANTA BERTIPE

Konstanta bertipe (*typed constant*) merupakan suatu konstanta yang selain ditunjukkan nilainya juga dinyatakan tipe datanya. Berbeda dengan konstanta yang telah dijelaskan sebelumnya, yang disebut dengan konstanta tak bertipe (*untyped constant*). Bentuk umum dari konstanta bertipe adalah sebagai berikut :

```
Const pengenal : tipe = konstanta;
```

dengan *pengenal* : nama konstanta
Tipe : tipe data konstanta
Konstanta : nilai konstanta

Konstanta dapat digunakan seperti halnya peubah dengan tipe yang sama, dan dapat muncul pada sisi kiri dari suatu statemen pemberian. Perhatikan bahwa konstanta bertipe ini nilainya diinisialisasikan hanya sekali, yakni di awal program. Sehingga, untuk setiap masukan ke dalam prosedur atau fungsi, konstanta bertipe lokal tidak akan diinisialisasikan kembali.

Konstanta bertipe sesungguhnya merupakan suatu peubah yang diberi nilai awal, maka tidak dapat saling dipertukarkan dengan konstanta biasa. Jika tipe konstanta yang digunakan termasuk dalam tipe terstruktur, maka deklarasinya akan menunjukkan nilai setiap komponennya. Dalam hal ini, Turbo Pascal mendukung konstanta bertipe array, record, set, dan pointer. Konstanta bertipe file, dan konstanta bertipe array dan record yang komponennya bertipe file tidak diperbolehkan.

UNGKAPAN

Ungkapan tersusun atas sejumlah operator dan operand. Kebanyakan operator Pascal bersifat biner, yang artinya adalah bahwa untuk setiap operator pasti digunakan oleh dua buah operand, sisanya bersifat operator tunggal (*unary*) yang hanya memerlukan sebuah operand. Operator tunggal selalu ditulis sebelum operand, misalnya –B. Dalam ungkapan yang lebih rumit, perlu untuk menentukan urutan operasi agar kita yakin atas hasil akhir yang akan diberikan, untuk itu diperlukan adanya urutan operasi. Aturan-aturan dasar mengenai urutan operasi sebagai berikut :

- Operand yang terletak di antara dua buah operator yang memiliki urutan berbeda akan dikerjakan terlebih dahulu sesuai dengan operator yang memiliki urutan yang lebih tinggi.
- Operand yang terletak di antara dua buah operator yang memiliki urutan sama, akan dikerjakan menurut operator yang berada di sebelah kirinya.
- Ungkapan yang terletak di dalam tanda kurung akan dikerjakan terlebih dahulu sebelum dianggap sebagai sebuah operand.

Urutan operasi pada ungkapan diberikan pada tabel sebagai berikut :

Operator	Urutan	Kategori
@, not	Pertama (tertinggi)	Operator tunggal
*, /, div, mod, and, shl, shr	Kedua	Operator pengali
+, -, or, xor	Ketiga	Operator penjumlahan
=, <>, <, >, >=, <=, in	Keempat (terendah)	Operator relasi

Seluruh operasi yang menggunakan urutan operasi yang sama selalu dikerjakan dari kiri ke kanan, meskipun kompiler mungkin memilih cara lain untuk mengoptimalkan pembangkitan kode.

1. Sintaksis Ungkapan

Ungkapan terbentuk dari faktor, suku, dan ungkapan-ungkapan sederhana. Faktor adalah operand yang digunakan untuk membentuk suatu ungkapan. Contoh faktor sebagai berikut :

```
X                          { nama peubah }
@X                         { pointer ke suatu peubah }
123                        { konstanta tak bertanda }
(A + B + C)                { sub ungkapan }
sin(A/2)                   { pemanggil fungsi }
[ '0' .. '9', 'A'..'Z']    { pembentuk himpunan }
not salah                  { ingkaran data bertipe boolean }
char(Angka+48)             { membentuk karakter ASCII }
```

Dua buah faktor atau lebih yang dioperasikan menggunakan operator pengali disebut dengan suku (*term*). Contohnya :

```
Jumlah/2
Awal*2
Salah or selesai
(X>=1000) and (X<1500)
```

Ungkapan sederhana adalah dua atau lebih suku yang dioperasikan menggunakan operator penjumlahan dan tanda bilangan. Contohnya :

```
Awal + Akhir - Rata2
Rata * Cacah + 1
```

Ungkapan kompleks adalah sejumlah ungkapan sederhana yang dioperasikan menggunakan operasi relasi, contohnya :

```
Rata = Jumlah / N
(I < J) = (J < K)
Angka in Digit
Salah <> Selesai
```

2. Operator

a. Operator Aritmatika

Tabel-tabel berikut ini menunjukkan operator-operator dan hasil-hasilnya untuk operasi aritmatika tunggal dan biner.

Operator	operasi	Tipe operand	Tipe hasil
+	Identitas	Integer, real	Integer, real
-	Ingkaran	Integer, real	Integer, real

Operator aritmatika tunggal

Operator	operasi	Tipe operand	Tipe hasil
+	Penjumlahan	Integer, real	Integer, real
-	Pengurangan	Integer, real	Integer, real
*	Perkalian	Integer, real	Integer, real
/	Pembagian	Integer, real	Integer, real
div	Pembagian	Integer	Integer
mod	Sisa pembagian	Integer	integer

Operator aritmatika biner

b. Operator Logika

Tipe operand untuk operasi logika sebagai berikut :

Operator	operasi	Tipe operand	Tipe hasil
Not	Ingkaran bit	Integer	Integer
And	Logika *And* bit	Integer	Integer
Or	Logika *Or* bit	Integer	Integer

Operator	operasi	Tipe operand	Tipe hasil
Xor	Logika *Xor* bit	Integer	Integer
Shl	Geser ke kiri	Integer	Integer
Shr	Geser ke kanan	integer	Integer

Operator logika

Pada tabel di atas, operator not termasuk operator *unary*. Dengan operator-operator ini, semua operand akan dikonversikan menjadi bilangan biner dan kemudian dioperasikan sesuai dengan operatornya.

c. Operator Boolean

Operator boolean digunakan untuk mengoperasikan operad-operand yang bertipe boolean. Untuk menentukan hasil operasi menggunakan operator ini, Turbo Pascal sering menggunakan dua buah model kode yang berbeda yaitu menggunakan perhitungan lengkap dan penghitungan menggunakan model *short-circuit*. Pada tabel di bawah ini, operator not termasuk operator *unary*.

Operator	operasi	Tipe operand	Tipe hasil
Not	Ingkaran	Boolean	Boolean
And	Logika *And*	Boolean	Boolean
Or	Logika *Or*	Boolean	Boolean
Xor	Logika *Xor*	Boolean	Boolean

Operator boolean

d. Operator Untai

Operator ini digunakan khusus untuk data yang bertipe string, atau packed string. Kegunaan dari operator untai adalah untuk menggabungkan (*concatenation*) daru dua atau lebih data untai. Dalam hal ini, hasil penggabungan tidak boleh lebih dari 255 karakter. Jika lebih dari 255 karakter, hasilnya akan terpotong menjadi 255 karakter saja.

Operator	operasi	Tipe operand	Tipe hasil
+	Penggabungan	String, char, atau packed string	String

Operator untai

e. Operator Himpunan

Operator-operator yang digunakan dalam operator himpunan apabila digunakan pada data yang bertipe himpunan (*set*) akan menghasilkan himpunan yang lain. Dalam hal ini, perlu kita perhatikan bahwa hasil operasi menggunakan data yang bertipe himpunan sesuai dengan aturan dalam himpunan logika :

- Nilai ordinal c ada dalam a + b, hanya jika c terdapat dalam a atau b.
- Nilai ordinal c ada dalam a – b, hanya jika c terdapat a dan tidak terdapat dalam b.
- Nilai ordinal c ada dalam a * b, hanya jika c terdapat dalam a dan b.

Operator	operasi	Tipe operand
+	Union	Tipe himpunan yang kompatibel
-	Selisih	Tipe himpunan yang kompatibel
*	Interseksi	Tipe himpunan yang kompatibel

Operator himpunan

f. Operator Relasi

Operator ini digunakan untuk membandingkan dua buah operand yang akan menghasilkan data bertipe boolean yang menunjukkan apakah pembandingan tersebut benar atau salah.

Operator	operasi	Tipe operand	Tipe hasil
=	Sama dengan	Sederhana, pointer, himpunan, string, packed string.	Boolean

Operator	operasi	Tipe operand	Tipe hasil
<>	Tidak sama dengan	Sederhana, pointer, himpunan, string, packed string.	Boolean
<	Lebih kecil dari	Sederhana, pointer, string, packed string.	Boolean
>	Lebih besar dari	Sederhana, pointer, string, packed string.	Boolean
<=	Lebih kecil atau sama dengan	Sederhana, pointer, string, packed string.	Boolean
>=	Lebih besar atau sama dengan	Sederhana, pointer, string, packed string.	Boolean
<=	Subset dari	Himpunan	Boolean
>=	Superset dari	Himpunan	Boolean
in	Anggota dari	*Operand kiri* : sembarang tipe ordinal T. *Operand kanan* : himpunan yang basisnya kompatibel dengan T.	Boolean

Operator relasi

3. Statemen

Statemen dapat dikatakan sebagai satuan terkecil suatu program. Statemen diawali dengan label yang nantinya digunakan sebagai titik acuan statemen goto. Statemen dapat dikelompokkan menjadi dua kelompok, yaitu statemen sederhana dan statemen terstruktur.

a. Statemen Sederhana

1) Statemen Pemberian

Statemen pemberian digunakan untuk mengubah nilai suatu perubah dengan nilai baru atau untuk menentukan suatu ungkapan yang nilainya dapat diperoleh dari fungsi yang digunakan. Contohnya :

```
X:= Y + Z;
Selesai := (I>=1) and (I<100);
Sinus :=sin(X/Y);
```

2) Statemen Prosedur

Statemen prosedur adalah statemen yang digunakan untuk memanggil suatu prosedur. Jika prosedur yang dipanggil berisi sederetan parameter formal, maka dalam statemen prosedur juga harus berisi parameter aktual yang sesuai (parameter dalam deklarasi prosedur disebut parameter formal, dan dalam statemen prosedur atau pemanggil prosedur disebut dengan parameter aktual). Contohnya

```
INVERS_MATRIX(Matrix_A, Matrix_B);
BUAT_LIST(Awal_List, Akhir_List, Data);
CETAK_DAFTAR_MENU;
```

3) Statemen Goto

Statemen goto digunakan untuk melompatkan proses ke suatu titik tertentu yang dinyatakan dalam label. Aturan yang harus diikuti dalam pemakaian goto sebagai berikut :

- Label yang akan dituju harus berada dalam blok yang sama. Dengan kata lain, tidak diperbolehkan untuk melompat keluar dari prosedur atau fungsi.

- Melompat masuk ke dalam statemen terstruktur dapat menyebabkan suatu pengaruh yang tidak terdefinisikan, meskipun kompiler tidak akan menunjukkan adanya kesalahan.

Contoh statemen goto sebagai berikut :

```
GOTO CETAK_LABEL;
```

Nama CETAK_LABEL yang akan digunakan sebelumnya sudah harus sudah didefinisikan dalam deklarasi label.

b. Statemen Terstruktur

1) Statmen Majemuk

Statemen majemuk (*compound statement*) merupakan statemen yang terdiri atas sejumlah statemen yang akan dieksekusi dengan urutan yang sama dengan urutan cara penulisan statemen-statemen tersebut. Komponen statemen ini diperlakukan sebagai satu statemen. Statemen majemuk ditandai dengan kata begin dan diakhiri dengan kata end. Contohnya :

```
Begin
   Bantu:=satu;
   Satu :=dua;
   Dua:=bantu;
End.
```

2) Statemen Kendali

a) Statemen If – Then – Else

Statemen if – then – else akan mencetak suatu kondisi dan menentukan apakah kondisi tersebut benar atau salah, kemudian melakukan sesuatu kegiatan sesuai dengan nilai kondisi tersebut. Bentuk umum dari statemen if – then – else sebagai berikut :

`If kondisi then statemen1;` atau

`If kondisi then statemen1 else statemen2;`

Dengan *kondisi* : kondisi yang dites untuk menentukan apakah *statemen1* atau *statemen2* yang akan dikerjakan.

Statemen1 : statemen yang akan dikerjakan jika kondisi bernilai true.

Statemen2 : statemen yang akan dikerjakan jika kondisi bernilai false.

b) Statemen Case

Statemen case berisi ungkapan (pemilihan) dan sederetan statemen, yang masing-masing diawali dengan satu atau lebih konstanta (disebut dengan konstanta case) atau dengan kata else. Pemilihan harus bertipe ordinal, sehingga tipe string, bilangan bulat yang bertipe longint atau word tidak dapat digunakan sebagai pemilihan. Seluruh konstanta case harus unik dan tipe ordinal yang digunakan harus sesuai dengan tipe pemilihan. Bentuk umum dari statemen case adalah :

```
Case pemilihan of
   Konstanta1 : statemen1;
   Konstanta2 : statemen2;
   .
   .
   .
```

```
    else : statemen_n;
end;
```

dengan *pemilihan* : nama peubah sebagai pemilih

konstanta1, konstanta2, ... : kemungkinan-kemungkinan nilai pemilihan.

Statemen1, statemen2, ... : statemen yang akan dikerjakan sesuai dengan nilai pemilihan.

3) Statemen Perulangan

a) Statmen Repeat .. Until

Statemen repeat digunakan sebagai awal statemen dan until sebagai akhir statemen yang dikerjakan berulang sekaligus untuk mengecek apakah proses berulang tersebut masih dapat diteruskan atau sudah harus berakhir. Bentuk umum dari statemen tersebut sebagai berikut :

```
Repeat statemen until kondisi;
```

Dengan *statemen* : statemen yang akan dikerjakan berulang.
Kondisi : syarat agar proses berulang dihentikan.

b) Statemen While .. do

Statemen while .. do hampir sama fungsinya dengan statemen sebelumnya (repeat .. until), dengan sedikit perbedaan bahwa dalam statemen while .. do kondisi untuk melakukan proses berulang dites di awal proses berulang, sedangkan pada statemen sebelumnya (repeat .. until) ada pada bagian akhir. Dengan demikian terlihat bahwa dalam statemen repeat .. until suatu statemen paling sedikit akan diproses satu kali, sedangkan dalam statemen while belum tentu. Bentuk umum dari statemen while sebagai berkut :

```
While kondisi do statemen;
```

Dengan *kondisi* : syarat agar proses berulang dapat berlangsung.

Statemen : statemen yang akan diproses berulang.

c) Statemen For

Statemen **for**, seperti halnya pada statemen **repeat .. until** dan statemen **while**, juga digunakan untuk melakukan proses berulang. Proses berulang dalam statemen for langsung dikendalikan oleh suatu peubah yang disebut dengan peubah kendali (*control variable*), yang harus bertipe ordinal. Bentuk umum statemen for sebagai berikut :

```
For peubah := awal to akhir do statemen;
```
atau
```
For peubah := akhir downto awal do statemen;
```

Dengan *peubah* : nama peubah kendali
Awal : nilai awal peubah kendali
Akhir : nilai akhir peubah kendali
Statemen : statemen yang akan diproses berulang

d) Statemen With

Statemen with banyak digunakan dalam manipulasi data yang bertipe record, yaitu untuk lebih mempersingkat cara penulisan medan-medan dari record.

4. Prosedur dan Fungsi

Prosedur dan fungsi memungkinkan untuk menambahkan sekelompok statemen yang seolah-olah terpisah dari program utama tetapi sesungguhnya merupakan bagian dari program utama. Prosedur diaktifkan menggunakan statemen prosedur (pemanggil prosedur) dan fungsi diaktifkan dengan suatu ungkapan yang hasilnya akan dikembalikan lagi sebagai nilai baru dari ungkapan tersebut.

a. Prosedur

Seperti telah dijelaskan sebelumnya, prosedur memiliki struktur yang sama dengan struktur program, yaitu terdiri atas nama prosedur, deklarasi-deklarasi dan bagian utama prosedur itu sendiri. Di dalam prosedur juga dimungkinkan adanya prosedur lain yang strukturnya sama. Bentuk ini dinamakan dengan prosedur tersarang (*nested procedure*).

Semua deklarasi dalam prosedur (label, konstanta, tipe data, dan peubah) dikatakan sebagai deklarasi lokal, sehingga hanya dapat digunakan dalam prosedur itu sendiri saja dan tidak dikenal di luar prosedur. Sedangkan deklarasi-deklarasi dalam program utama bersifat global, sehingga dapat digunakan dalam bagian program yang manapun. Bentuk umum dari deklarasi prosedur sebagai berikut :

```
Procedure nama <(daf_par)>;
```

Dengan *nama* : nama prosedur
Daf_par : daftar parameter formal

Parameter formal ada dua macam, yaitu parameter nilai (*value parameter*) dan parameter pubah (*variable parameter*). Parameter nilai adalah parameter yang tidak diawali dengan kata baku *var*, dan parameter peubah adalah parameter yang diawali dengan kata baku *var*.

Statemen prosedur terdiri atas dua bagian, yaitu nama prosedur yang akan dipanggil dan daftar parameter formal. Urutan parameter dalam daftar parameter harus sesuai dengan urutan parameter dalam parameter formal pada deklarasi prosedur, terutama dalam hal tipe datanya. Pada saat prosedur dipanggil, maka semua nilai parameter dalam daftar parameter aktual diberikan pada parameter dalam daftar parameter formal dengan urutan yang sesuai.

b. Fungsi

Secara umum, fungsi hampir sama dengan prosedur, dengan sedikit perbedaan bahwa nama fungsi sekaligus berfungsi sebagai suatu peubah, sehingga dalam deklarasi fungsi harus dinyatakan tipe datanya. Bentuk umum dari deklarasi fungsi sebagai berikut :

```
Function nama <(daf_par)> : tipe;
```

Dengan *nama* : nama fungsi
Daf_par : daftar parameter formal
Tipe : tipe data dari fungsi tersebut

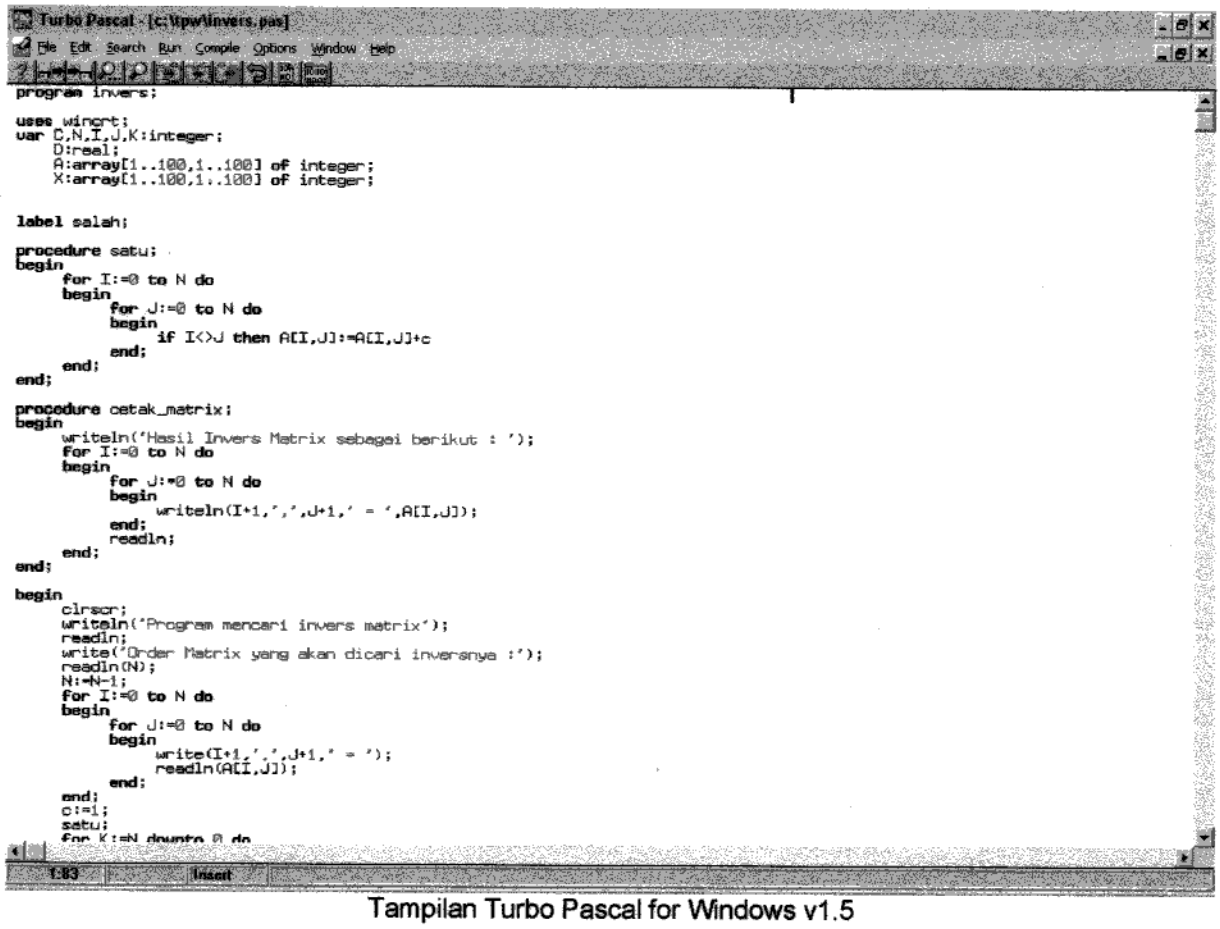
Dalam fungsi, semua parameter formal harus berupa parameter nilai, tidak diperbolehkan ada parameter peubah.

c. Deklarasi Forward

Seperti telah dijelaskan sebelumnya, prosedur atau fungsi sudah harus dituliskan terlebih dahulu sebelum digunakan. Tetapi ada kalanya kita memanggil suatu prosedur dimana prosedur tersebut ditempatkan sesudah bagian yang memanggilnya tadi. Untuk itu harus digunakan deklarasi ke depan (*forward*).

Dalam deklarasi *forward*, pada prosedur yang dimaksud harus digunakan kata baku forward, yang dapat diikuti dengan parameter untuk prosedur tersebut. Baris berikutnya adalh prosedur yang lain. Sedangkan tubuh prosedur yang di*forward*kan dapat ditulis di sembarang tempat tanpa perlu menuliskan lagi daftar parametemya. Dengan demikian, apabila kata forward digunakan, deklarasi prosedur dapat dibagi menjadi dua, yaitu bagian deklarasi forward dan bagian deklarasi prosedurnya sendiri. Kedua bagian ini membentuk satu prosedur secara utuh seperti halnya sebuah prosedur (atau fungsi) yang

dideklarasikan sendiri. Dengan cara ini dimungkinkan untuk membentuk *mutual recursive*, yaitu prosedur A akan memanggil prosedur B, sebaliknya dalam prosedur B juga terdapat statemen untuk memanggil prosedur A.



Tampilan Turbo Pascal for Windows v1.5

Contoh salah satu program untuk menghitung rerata nilai dalam bahasa Pascal. Perhatikan cara penulisannya.

```
program ArrayNilai;
uses wincrt;

const max_test=5;
type Daftar_Nilai=array[1..max_test] of byte;
var I          : byte;
    jumlah     : real;
    Nilai_tes  : Daftar_Nilai;

begin
     clrscr;
     {memasukkan data ke array}
     for I:=1 to 5 do
     begin
          write('Masukkan nilai tes ke ',I,' :');
          readln(Nilai_tes[I]);
     end;
{menghitung dan menampilkan nilai rata-rata }
jumlah:=0;
for I:=1 to 5 do
    jumlah:=jumlah+Nilai_tes[I];
    writeln('Rata-rata Nilai = ',Jumlah/5:6:1);
    readln;
end.
```

ARRAY

Array merupakan tipe data terstruktur yang berisi sekumpulan komponen/elemen dengan tipe sama. Elemen dari masing-masing array dapat diakes dengan menyebutkan nama array dan indeks array.

Definisi tipe array

Pendefinisian tipe array terdiri dari kata-tercadang **ARRAY** diikuti dengan ***<tipe_indeks>*** yang diletakkan dalam tanda kurung siku, kata tercadang **OF** dan kemudian ***<tipe_komponen>*** dan diakhiri dengan tanda titik-koma (;).

```
ARRAY [<tipe_indeks_1> .. <tipe_indeks_2>] OF <tipe_komponen>;
```

Jumlah ***<tipe_indeks>*** dapat terdiri lebih dari satu. Jika terdapat lebih dari satu tipe indeks, maka antar tipe indeks dipisahkan dengan tanda koma. Masing-masing tipe indeks dapat berbeda. Namun hanya terbatas pada tipe ordinal (selain longint dan subrange dari longint). Tipe seperti char, Boolean, ataupun enumerasi, diperkenankan. ***<Tipe_komponen>*** menyatakan tipe dari elemen yang terdapat dalam array dan dapat berupa sebarang tipe (asalkan bukan file). Contoh pendefinisian tipe array dan variabel array sebagai berikut :

```
Type
     Operator = (plus, minus, kali, bagi);
     Tanda    = Array[operator] of integer;
     ArrayHrf = Array['A'..'Z'] of byte;

Var
     JumlahTanda : Tanda;
     FrekHrf     : ArrayHrf;
     Daftar      : Array[1..3] of byte;
```

Pada contoh :

- ***Tanda*** adalah tipe array, dengan tipe indeks adalah enumerasi dan tipe komponen berupa integer.
- ***ArrayHrf*** adalah tipe array, dengan tipe indeks adalah char ('A' sampai dengan 'Z') dan tipe komponen berupa byte.
- ***JumlahTanda*** adalah variabel bertipe ***Tanda***.
- ***FrekHrf*** adalah variabeil bertipe ***ArrayHrf***.
- ***Daftar*** adalah variabel array dengan tipe indeks berupa 1, 2, dan 3. Adapun tipe komponennya adalah byte.

Array Berdimensi Satu

Array berdimensi satu merupakan array yang memiliki hanya satu tipe indeks dan komponennya bukan berupa array. Semua contoh di atas termasuk array berdimensi satu. Pengakesan elemen array berdimensi satu dilakukan dengan menggunakan bentuk :

```
Variabel_array[indeks_array];
```

Untuk mengisikan nilai 32 ke variabel Daftar (berdasarkan deklarasi di depan) yang berindeks sama dengan 1, pernyataan penugasannya berupa :

```
Daftar[1]:=32;
```

Adapun untuk menampilakn isi **Daftar[3]**, perintahnya adalah sebagai berikut :

```
WRITELN(Daftar[3]);
```

Walaupun ***Daftar*** pada contoh di depan dideklarasikan dengan nilai indeks berkisar antara 1 sampai dengan 3, pengaksesan array ***Daftar*** dengan nilai indeks yang lain (misalnya 4) tak akan ditolak oleh Turbo Pascal (kecuali kalau nilai indeks berupa konstanta). Namun hal tersebut harus dihindari, sebab dapat menimbulkan masalah (misalnya menyebabkan nilai variabel lain berubah). Hal ini ditunjukkan pada contoh sebagai berikut :

```
Program Array1;

{ ------------------------
  Efek indeks array
  Di luar kawasan deklarasi
  ------------------------- }

uses wincrt;

var
   Daftar   : array[1..3] of byte;
   Temp     : char;
   I        : byte;

Begin
      Temp:='A';          (* Isi semula *)
      WRITELN('Isi Temp semula : ',Temp);
      WRITELN('Masukkan empat buah bilangan bulat.');
      FOR I:=1 TO 4 DO
            READ(Daftar[I]);
      WRITELN('Isi array Daftar : ');
      FOR I:=1 TO 4 DO
            WRITELN(Daftar[I]);
      WRITELN('Isi Temp kini : ',Temp);
End.
```

Contoh eksekusi dari program di atas :

```
Isi Temp semula : A
Masukkan empat buah bilangan bulat.
77 88 99 67 ↵
Isi array Daftar :
77
88
99
67
Isi Temp kini : C
```

Pada contoh di atas, ***Daftar[4]*** diisi dengan **67**, dan saat ditampilkan ke layar ***Daftar[4]*** memang tetap bernilai **67**. Tetapi apa yang terjadi dengan isi variabel ***Temp***? Isi variabel ***Temp*** berubah karena efek pemberian nilai indeks pada ***Daftar*** di luar indeks yang telah ditentukan (yaitu antara 1 sampai dengan 3). Contoh pemakaian array berdimensi satu yaitu untuk menampung sejumlah nama orang. Kemudian nama-nama yang ada pada array diurutkan secara alfabetis. Program untuk keperluan tersebut adalah sebagai berikut :

```
Program Array2;

{ ----------------------------------------
  Contoh pengurutan nama secara alfabetis.
  Array dipakai untuk menampung sejumlah nama
  ---------------------------------------- }

Uses Wincrt;

Const
      MaksOrang = 20;

Type
      StringNama = STRING[20];

Var
      I,J,n : byte;
      Temp  : StringNama;
      Nama  : Array[1..MaksOrang] Of StringNama;

BEGIN
      Clrscr;
      (* Pemasukan data nama *)
      WRITE('Jumlah data : ');READLN(n);
      WRITELN;WRITELN('Data semula : ');
      WRITELN;
      FOR I:= 1 TO n DO
      Begin
            WRITE(I:2,'.');
            READLN(Nama[I]);
      End;
      (* Proses Pengurutan *)
      FOR I:= 1 TO n-1 DO
            FOR J:=1 TO n DO
                  IF Nama[I]>Nama[J] THEN
                  Begin
                        { ----- Penukaran Isi Array ----- }
                        Temp:=Nama[I];
                        Nama[I]:=Nama[J];
                        Nama[J]:=Temp;
                  End;
      { ------ Tampilkan Hasil Pengurutan ----- }
      WRITELN;
      WRITELN(' DATA SETELAH DIURUTKAN ');
      WRITELN;
      FOR I:= 1 TO n DO
            WRITELN(I:2,'.',Nama[I]);
END.
```

Cobalah untuk menuliskan program di atas, dan kemudian lihat apa yang terjadi.

Pada program di atas, metode yang digunakan untuk pengurutan yaitu bubble-sort. Contoh yang lain berupa pemakaian array dengan tipe indeks yaitu karakter. Array dipakai untuk mencatat banyaknya masing-masing huruf yang ada pada suatu kalimat.

```
Program Array3;

{ ---------------------------------------
  contoh array dengan indeks bertipe CHAR.
  Program digunakan untuk menghitung
  Cacah dari masing-masing karakter dalam
  Suatu kalimat.
  --------------------------------------- }

Uses Wincrt;
Const
      Spasi=#32;
      Tilde='~';
Type
      ArrayKarakter = Array[Spasi..Tilde] Of Byte;
Var
      I           : Byte;
      Kalimat     : String[80];
      FrekKar     : ArrayKarakter;
      Karakter    : Char;

Begin
      Clrscr;
      (* Seluruh array diberi nilai awal nol *)
      FOR Karakter:= Spasi TO Tilde DO
            FrekKar[Karakter]:=0;
      (* Pemasukan Kalimat *)
      WRITELN('Masukkan sembarang kalimat : ');
      READLN(Kalimat);
      (* Hitung cacah per karakter *)
      FOR I:= 1 TO LENGTH(Kalimat) DO
      BEGIN
            Karakter:=Kalimat[I];
            FrekKar[I]:=FrekKar[I]+1;
      END;
      (* Tampilkan cacah masing-masing karakter *)
      FOR Karakter:=Spasi TO Tilde DO
            IF FrekKar[Karakter]<>0 THEN
                  WRITELN('Jumlah Karakter ',Karakter,
                        ' = ', FrekKar[Karakter]:3);
End.
```

Jalankan program tersebut di atas, cek apakah ada kesalahan, kemudian lihat hasil eksekusinya.

Array Multidimensi

Array multidimensi adalah array yang memiliki lebih dari satu tipe indeks. Sebagai contoh :

```
Type
     Matriks = Array[1..3] of Array[1..5] of integer;
     TigaDim = Array[1..2] of Array[1..3] of Array[1..5] of Char;
```

- ***Matriks*** merupakan tipe yang mengandung 3 elemen, dengan masing-masing elemen terdiri atas 5 elemen bertipe integer. Dengan kata lain, matriks mempunyai 3x5 elemen bertipe integer.
- ***TigaDim*** merupakan tipe yang mengadung 2x3x5 elemen bertipe char.

Bentuk seperti ***Matriks*** dan ***TigaDim*** dapat disederhanakan menjadi :

```
Type
     Matriks = Array[1..3,1..5] of integer;
     TigaDim = Array[1..2,1..3,1..5] of char;
```

Tipe seperti ***Matriks*** dinamakan array berdimensi dua, sedangkan tipe ***TigaDim*** disebut array berdimensi tiga. Misalkan variabel X didefinisikan bertipe ***Matriks***, maka :

```
Var X : Matriks;
```

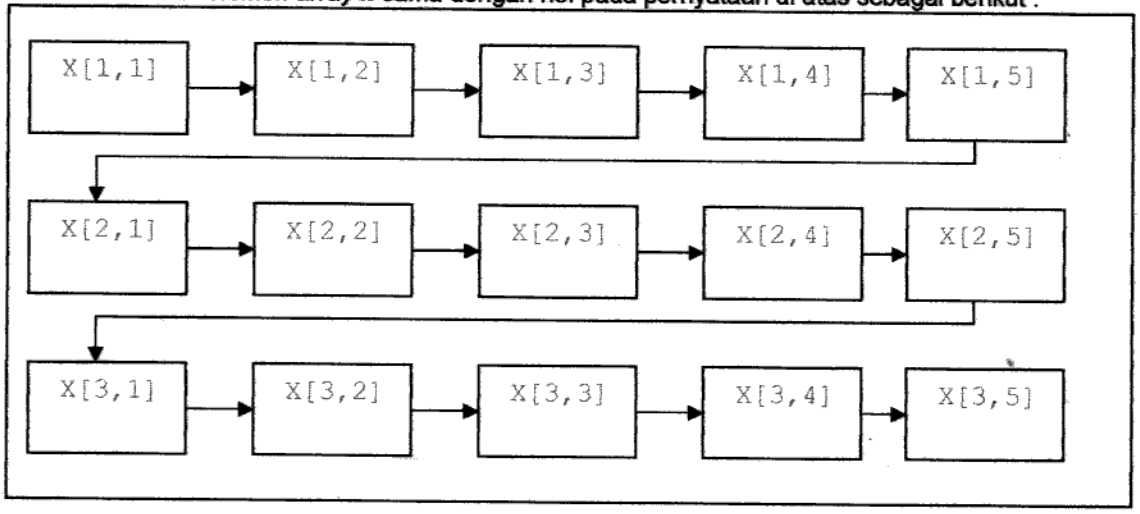
Maka, struktur array X dapat dilukiskan sebagai berikut :

	Kolom 1	Kolom 2	Kolom 3	Kolom 4	Kolom 5
Baris 1					
Baris 2					
Baris 3					

Setiap elemen dari array berdimensi dua diakses dengan menggunakan bentuk `X[I,J]` yang menyatakan elemen `X` pada baris `I` dan kolom `J`. Pada diagram di depan, elemen yang ditandai dengan arsiran memiliki identitas `X[2,3]`.
Untuk array berdimensi tiga, identitas dari suatu elemen array ditentukan oleh tiga buah indeks array. Contoh berikut ini menunjukkan cara membuat keseluruhan elemen array `X` bernilai nol :

```
FOR I:=1 TO 3 DO
     FOR J:=1 TO 5 DO
          X[I,J]:=0;
```

Proses membuat elemen array `X` sama dengan nol pada pernyataan di atas sebagai berikut :



contoh berikut ini memberikan gambaran cara memasukkan data matriks yang berasal dari keyboard ke dalam array berdimensi dua, dan juga menampilkan data matriks ke layar.

```
Program Array4;

{ ------------------------------
  Contoh memasukkan data matriks
  Ke dalam array berdimensi dua,
  Serta menampilkan isinya
  ------------------------------ }

Uses Wincrt;

Var
      I,J,Baris,Kolom : integer;
      X : Array[1..10,1.100] of real;

Begin
      Clrscr;
      (* Membaca data dan menempatkan ke array *)
      WRITE('Banyaknya baris (max 10) : ');READLN(Baris);
      WRITE('Banyaknya kolom (max 10) : ');READLN(Kolom);
      FOR I:= 1 TO Baris DO
      BEGIN
            FOR J:= 1 TO Kolom DO
                  BEGIN
                        WRITE('Elemen (',I,',',J,') = ');
                        READLN(X[I,J]);
                  END;
            WRITELN;
      END;

      (* Tampilkan elemen matriks *)
      WRITELN(' Matriks : ');WRITELN;
      FOR I:= 1 TO Baris DO
      BEGIN
            FOR J:= 1 TO Kolom DO
                  WRITE(X[I,J]:8:3);
                  WRITELN
      END;
End.
```

Pada contoh di atas, pernyataan

```
FOR J:= 1 TO Kolom DO
     WRITE(X[I,J]:8:3);
```

Menyebabkan semua elemen dari baris I dicetak dalam satu baris. Untuk memindahkan penampilan ke baris berikutnya, perintah yang digunakan yaitu : `WRITELN`.

Menyalin Array

Pada operasi array, kadangkala diperlukan untuk menyalin seluruh elemen array ke array lain. Misalnya saja ***Vektor*** dan ***Temporer*** merupakan variabel array yang dideklarasikan sebagai berikut :

```
Type
      Arr = Array[1..5] of Real;
Var
      Vektor,Temporer : Arr;
```

Bila nilai dari array ***Vektor*** hendak disalin ke ***Temporer***, maka pernyataan yang dipakai berupa :

```
FOR I:= 1 TO 5 DO
      TEMPORER[I]:=VEKTOR[I];
```

Penyalinan array semacam ini dapat dilaksanakan dengan memakai pernyataan yang lebih sederhana :

```
Temporer :=  Vektor;
```

Syarat berlakunya pernyataan seperti di atas adalah antara kedua variabel array harus memiliki tipe yang tepat sama. Bila misalnya dideklarasikan ;

```
Type
      Arr = Array[1..5] of real;
Var
      S : Arr;
      T : Array[1..5] of real;
```

Pernyataan seperti `S:=T` menyebabkan kesalahan sewaktu kompilasi, sekalipun `S` dan `T` sama-sama array yang mengandung 5 elemen bertipe real.

Soal

1. Apakah hasil eksekusi dari program berikut ini?

```
Program Soal_1;
Var
      I : byte;
      Y : Array[1..4] of Real;
Begin
      Y[1]:=5;
      Y[2]:=8;
      Y[3]:=9;
      Y[4]:=12;
      FOR I:= 4 DOWNTO 1 DO
      Begin
            Y[I]:=Y[I]-1;
            WRITELN(Y[I]);
      End;
End;
```

2. Buatlah program untuk memasukkan sejumlah data real ke dalam array berdimensi satu. Selanjutnya program memberikan informasi berupa :
   a. Nilai terbesar
   b. Nilai terkecil
   c. Nilai rata-rata

3. Buatlah program untuk memasukkan dua buah matriks berukuran n x n (n dimasukkan dari keyboard). Kemudian program menjumlahkan kedua matriks tersebut. Hasilnya ditampilkan di layar.
   Contoh :

$$\begin{bmatrix} 1 & 9 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 15 \\ 10 & 12 \end{bmatrix}$$

Persamaan Linear Simultan

Persamaan linear simultan dalam masalah rekayasa hampir merupakan bagian yang tidak terpisahkan dari cara analisis atau hitungan rumusan model matematika permasalahan. Diperkirakan 70% penyelesaian rumusan matematika dalam soal rekayasa mengambil bentuk persamaan linear. Kasus yang terpenting adalah jika jumlah besaran atau variable yang dicari sama jumlahnya dengan jumlah persamaan atau yang lazim disebut persamaan linear yang simultan.
Terdapat tiga metode untuk menyelesaikian persamaan linear simultan. Cara pertama disebut metode iterasi, cara kedua melalui proses eliminasi, dan yang ketiga dengan cara determinan. Tidaklah salau dapat ditentukan, dari ketiga metode tersebut, cara ap ayang lebih baik bagi penyelesaian persamaan linear. Aturan Cramer untuk metode determinan memberikan besaran eksak, tetapi metode ini menjadi sukar digunakan jika besaran yang dicari lebih dari 4 variabel. Untuk menyelesaikian system persamaan linear yang besar, dengan jumlah besaran yang dicari sangat banyak, umumnya digunakan metode iterasi atau eliminasi.

1. Metode Eliminasi

Penyelesaian secara eliminasi untuk sistem persamaan linear simultan merupakan proses eliminasi bertahap dari variabel yang dicari pada persamaan simultan sampai diperoleh bentuk persamaan dengan satu variabel. Secara umum bentuk persamaan linear simultan adalah sebagai berikut :

$$\begin{aligned}
A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + \ldots + A_{1n}X_n &= B_1 \\
A_{21}X_1 + A_{22}X_2 + A_{23}X_3 + \ldots + A_{2n}X_n &= B_2 \\
A_{31}X_1 + A_{32}X_2 + A_{33}X_3 + \ldots + A_{3n}X_n &= B_3 \\
&\ldots \\
A_{n1}X_1 + A_{n2}X_2 + A_{n3}X_3 + \ldots + A_{nn}X_n &= B_n
\end{aligned}$$

Atau dalam notasi matrik :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ . & . & . & \ldots & \ldots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ b_n \end{Bmatrix}$$

dengan memberikan data a_{ij} dan nilai b_i pada bentuk persamaan linear simultan, maka penentuan harga x_i dapat dilakukan dengan beberapa prosedur eliminasi. Beberapa cara dijelaskan berikut ini :

a. Metode Eliminasi Gauss

Gauss menyelesaikan persamaan linear simultan dengan melalui proses menghilangkan atau mengganti secara beruntun beberapa besaran yang dicari samapi sistem menjadi satu persamaan dengan satu besaran.
Persamaan yang menyatakan satu variabel yang tidak diketahui disebut persamaan **PIVOTAL** atau persamaan **POROS**. Jika telah diketahui nilai satu variabel, yaitu persamaan yang mempunyai koefisien terbeasr dari besaran yang akan dieliminasi.
Apabila besaran yang akan dieliminasi secara berturut-turut adalah x_1, x_2, maka persamaan pivotal pertama diperoleh dari koefisien x_1 mutlak yang terbesar. Persamaan ini dipindahkan posisinya pada susunan baris pertama, sehingga koefisien yang terbesar berada pada lokasi diagonal a_{11}. Persamaan pivotal kedua x_2 dari hasil susunan persamaan pivotal pertama dipilih dari koefisien besaran x_2 yang terbesar. Demikian seterusnya sehingga tersusun persamaan linear simultan dengan koefisien diagonal dapat ditulis sebagai berikut :

$$\begin{aligned}
A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + \ldots + A_{1n}X_n &= B_1 \\
A_{21}X_1 + A_{22}X_2 + A_{23}X_3 + \ldots + A_{2n}X_n &= B_2, \text{ hal mana } |a_{ii}| \geq |a_{ij} + \ldots + a_{i,i-1} + a_{i,i+1}, a_{in}| \\
A_{31}X_1 + A_{32}X_2 + A_{33}X_3 + \ldots + A_{3n}X_n &= B_3
\end{aligned}$$

Tinjaulah contoh penyelesaian sistem persamaan linear dengan cara Gauss sebagai berikut. Persamaan linear dengan 4 besaran yang tidak diketahui disusun pada empat persamaan :

(1) $2.63x_1 + 5.21x_2 - 1.694x_3 + 0.938x_4 - 4.230 = 0$
(2) $3.16x_1 - 2.95x_2 + 0.813x_3 - 4.210x_4 + 0.716 = 0$
(3) $5.36x_1 + 1.88x_2 - 2.150x_3 - 4.950x_4 - 1.280 = 0$
(4) $1.34x_1 + 2.98x_2 - 0.432x_3 - 1.768x_4 - 0.419 = 0$

besaran yang akan dihilangkan berturut-turut adalah x_1, x_2, dan x_3. karena koefisien x_1 terbesar pada persamaan (3), ini merupakan persamaan pivotal pertama. Nyatakan x_1 dari persamaan ini :

(5) $$x_1 = -\frac{1.88}{5.36}x_2 + \frac{2.150}{5.36}x_3 + \frac{4.950}{5.36}x_4 + \frac{1.280}{5.36}$$
$= -0.350746x_2 + 0.401119x_3 + 0.923506x_4 + 0.238806$

masukkan nilai x_1 ini kedalam persamaan (1), (2), dan (4), sehingga diperoleh :
(6) $4.287538x_2 - 0.63906x_3 + 3.36682x_4 - 3.601940 = 0$
(7) $-4.058360x_2 + 2.08054x_3 - 1.29172x_4 + 1.470627 = 0$
(8) $2.510000x_2 + 0.10550x_3 - 0.53050x_4 - 0.099000 = 0$

dari persamaan (6), (7), dan (8) terlihat koefisien terbesar x_2 pada persamaan (6), sehingga ini merpuakan persamaan pivotal kedua. Nyatakan x_2 dari persamaan ini :
(9) $x_2 = 0.149051x_3 - 0.785258x_4 + 0.840096$

isikan (9) pada (7) dan (8) sebagai berikut :
(10) $1.475640x_3 + 1.89514x_4 - 1.93879 = 0$
(11) $0.479618x_3 - 2.50150x_4 + 2.00964 = 0$

persamaan (10) adalah persamaan pivotal ketiga, sehingga :
(12) $x_3 = -1.28428x_4 + 1.31386$

isikan (12) ke (11) sehingga diperoleh :
(13) $-0.3117463x_4 + 2.63979 = 0$

yang memberikan nilai untuk $x_4 = 0.846775$.
Dengan cara substitusi ke belakang, besaran x_3, x_2, dan x_1 diperoleh dari persamaan (12), (9), dan (5). Dengan mengisikan nilai x_4 ke persamaan (12) diperoleh $x_3 = 0.22636$ dan dengan mbisikan nilai x_3 dan x_4 ke persamaan (9), diperoleh $x_2 = 0.208898$, sehingga nilai x_1 diperoleh dari persamaan (5) dari x_2,x_3, dan x_4 yang telah diketahui, yaitu $x_1 = 1.038335$. hasil penyelesaian sistem yaitu :
$X_1 = 1.038335$
$X_2 = 0.208898$
$X_3 = 0.22636$
$X_4 = 0.846775$

Untuk memeriksa kebenaran keempat nilai di atas, masukkan nilai x1, x2, x3, dan x4, pada persamaan (1), (2), (3), dan (4). Terlihat bahwa cara Gauss menyelesaikan sistem persamaan linear adalah denganmengubah sistem persamaan yang diketahui menjadi persamaan pivotal sistem triangulasi. Dalam penyajian matriks, susunan akhir menjadi :

$$\begin{bmatrix} 5.36 & 1.880000 & -2.15000 & -4.95000 \\ 0 & 4.287538 & -0.63906 & 3.36682 \\ 0 & 0 & 1.47564 & 1.89514 \\ 0 & 0 & 0 & -3.117463 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.28000 \\ 3.60194 \\ 1.93897 \\ -2.63979 \end{Bmatrix}$$

Dengan mudah dari matriks ini, dihitung determinannya, berupa perkalian nlai koefisien diagonal utama : D = 5.36 x 4.287538 x 1.47564 x (- 3.117463) = -105.720. Pada contoh terlihat bahwa persamaan awal mengalami proses transformasi matriks dasar secara bertahap mulai dari susunan awal :

$$\begin{bmatrix} 2.63 & 5.21 & -1.694 & 0.938 \\ 3.16 & -2.95 & 0.813 & -4.210 \\ 5.36 & 1.88 & -2.150 & -4.950 \\ 1.34 & 2.98 & -0.432 & -1.768 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 4.230 \\ -0.716 \\ 1.280 \\ 0.419 \end{Bmatrix}$$

ke susunan persamaan pivotal pertama

$$\begin{bmatrix} 5.36 & 1.88 & -2.15 & -4.95 \\ 2.63 & 5.21 & -1.69 & 0.94 \\ 3.16 & -2.95 & 0.81 & -4.21 \\ 1.34 & 2.98 & -0.43 & -1.77 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.280 \\ 4.230 \\ -0.716 \\ 0.419 \end{Bmatrix}$$

dari baris pertama, dengan menyatakan nilai $x_1 = f(x_2, x_3, x_4)$ dan memasukkannya ke baris ke-2, ke-3, dan ke-4, diperoleh susunan :

$$\begin{bmatrix} 5.36 & 1.88 & -2.15 & -4.95 \\ 0 & 4.29 & -0.64 & 3.37 \\ 0 & -4.06 & 2.08 & -1.29 \\ 0 & 2.51 & 0.11 & -0.53 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.280 \\ 3.602 \\ -1.471 \\ 0.099 \end{Bmatrix}$$

untuk persamaan pivoting kedua, tidak perlu perubahan susunan ,baris karena koefisien variabel x_2, yaitu $|4.29| \geq |-0.64 + 3.37|$, sudah terpenuhi. Dari baris kedua ini ,dengan menyatakan $x_2 = f(x_3, x_4)$, dan mengisikannya ke dalam baris ke-3 dan ke-4 diperoleh susunan :

$$\begin{bmatrix} 5.36 & 1.88 & -2.15 & -4.95 \\ 0 & 4.29 & -0.64 & 3.37 \\ 0 & 0 & 1.48 & 1.90 \\ 0 & 0 & 0.48 & -2.50 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.280 \\ 3.602 \\ 1.939 \\ -2.010 \end{Bmatrix}$$

dari susunan ini, dengan menyatakan x3 dalam fungsi x4, dan memasukkan nilai ini ke baris ke-4 diperoleh susunan :

$$\begin{bmatrix} 5.36 & 1.88 & -2.15 & -4.95 \\ 0 & 4.29 & -0.64 & 3.37 \\ 0 & 0 & 1.48 & 1.90 \\ 0 & 0 & 0 & -3.12 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.280 \\ 3.602 \\ 1.939 \\ -2.640 \end{Bmatrix}$$

dengan penjelasan dalam contoh di atas, Algoritma program mengubah bentuk umum persamaan simultan :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ . & . & . & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ b_n \end{Bmatrix}$$ menjadi bentuk matriks triangulasi

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ 0 & A_{22} & A_{23} & A_{24} \\ 0 & 0 & A_{33} & A_{34} \\ 0 & 0 & 0 & A_{44} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix}$$

Hal ini dilakukan melalui proses men-nol-kan kolom 1 sampai dengan kolom n-1 di bawah posisi diagonal. Untuk tujuan ini dibutuhkan (n-1) tahapan proses. Setiap tahap k, k = 1, 2, ..., n-1 akan menghasilkan nilai 0 padak kolom k tanpa mengubah niali 0 yang sudah ada pada kolom sebelumnya. Ini berarti bahwa pada setiap tahap dicari suatu pengali m_{ik}, dan kemudian dilakukan pengurangan hasil pengali dari baris persamaan pivoting yang ditinjau dengan persamaan dari baris lainnya sedemikian rupa sehingga diperoleh nilai nol. Untuk mendapatkan nilai nol pada kolom pertama di bawah diagonal elemen a_{11}, pada contoh berikut :

$$\begin{bmatrix} 5.36 & 1.88 & -2.15 & -4.95 \\ 2.63 & 5.21 & -1.69 & 0.94 \\ 3.16 & -2.95 & 0.81 & -4.21 \\ 1.34 & 2.98 & -0.43 & -1.77 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1.280 \\ 4.230 \\ -0.716 \\ 0.419 \end{Bmatrix}$$

dihitung : $m_{21} = \dfrac{a_{21}}{a_{11}} = \dfrac{2.63}{5.63} = 0.467$

$a_{21\ baru} = a_{21} - m_{21}.a_{11} = 2.63 - 0.467*5.36 = 0$
$a_{22\ baru} = a_{22} - m_{21}.a_{12} = 5.21 - 0.467*1.88 = 4.33$
$a_{23\ baru} = a_{23} - m_{21}.a_{13} = - 1.69 + 0.467*2.15 = - 0.69$
$a_{24\ baru} = a_{24} - m_{21}.a_{14} = - 0.94 + 0.467*4.95 = 3.25$
$b_{2\ baru} = b_2 - m_{21}.b_1 = 4.23 - 0.467*1.28 = 3.63$

$$m_{31} = \frac{a_{31}}{a_{11}} = \frac{3.16}{5.63} = 0.561$$

$a_{31\ baru} = a_{31} - m_{31}.a_{11} = 3.16 - 0.561*5.36 = 0$
$a_{32\ baru} = a_{32} - m_{31}.a_{12} = - 2.95 - 0.561*1.88 = -4.00$
$a_{33\ baru} = a_{33} - m_{31}.a_{13} = 0.81 + 0.561*2.15 = 2.02$
$a_{34\ baru} = a_{34} - m_{31}.a_{14} = - 4.21 + 0.561*4.95 = 3.25$
$b_{3\ baru} = b_3 - m_{31}.b_1 = - 0.716 - 0.561*1.28 = -1.43$

secara umum :

$$m_{ik} = \frac{a_{ik}}{a_{kk}}$$

$a_{ij\ baru} = a_{ij} - m_{ik}.a_{kj}$, j = 1,2,...,n
$b_{i\ baru} = b_i - m_{ik}.b_k$, untuk i = k+1, k+2, ..., n dan k = 1,2,..., n-1
Pada proses perhitungan besaran ini sesungguhnya hanya ditijau nilai j = k+1, k+2, ..., n, karena besaran nol di bawah posisi diagonal tidak memerlukan perhitungan lanjut. Dengan substitusi ke belakang :

$$x_n = \frac{b_n}{a_{nn}} \text{ dan } x_1 = \left(\frac{b_i - a_{i,i+1}x_{i+1} - a_{i,i+2}x_{i+2} - \ldots - a_{in}x_n}{a_{11}} \right)$$

dengan i = n-1, n-2, n-3,...,2,1, akan diperoleh besaran variabel yang dicari.
Elemen a_{kk} yang digunakan menghitung m_{ik} disebut elemen PIVOT. Pada tahap akhir penghitungan, determinan dinyatakan sebagai $|A| = \prod_{i-1}^{n} a_{ii}$ sehingga jika tidak ada pivot

bernilai nol, beratri determinan | A | = 0. Ini menunjukkan invers $[A]^{-1}$ tidak ada, dan tidak ada penyelesaian unik persamaan sebab solusi vektor x dicari dari $\{x\} = [A]^{-1}\{b\}$.
Jika pada proses eliminasi nilai a_{kk} bernilai 0, tetapi elemen di bawahnya bukan 0, maka perlu modifikasi susunan baris, dengan penukaran baris dalam matriks untuk mendapatkan pivot yang bukan bernilai 0. proses ini disebut proses PIVOTING.
Suatu pivot bernilai kecil sekali, dan sistem persamaan mempunyai nilai determinan yang kecil; sistem tersebut disebut berkondisi *ill* (*ill conditioned*); yang berarti solusi yang akan diperoleh tidak memberikan hasil yang besar. Penjelasan uraian ini dapat dilihat pada solusi dua persamaan berikut ini :

$0.10x_1 + 0.80x_2 = 11.0$
$0.09x_1 + 0.81x_2 = 11.4$

yang dalam penyajiannya secara grafik hampir paralel. Solusi persamaan ini tidak stabil dan hasilnya dengan cara apapun tidak akan memberikan nilai yang benar. Algoritma yang memberikan sifat tidak stabil harus dicegah dengan menetapkan syarat perlu :

$|m_{ik}| \le 1$, yaitu $|a_{kk}| \ge \left|\sum_{i=k+1}^{k} a_{ik}\right|$ pada tahap k, dengan i = k+1, ... , n

Dengan ketentuan ini, prosedur pivoting perlu di modifikasi pada tahap ke-k, sebelum dibentuk pengali m_{ik} dengan penyusunan baris baru sedemikian rupa untuk memperoleh nilai mutlak terbesar elemen dalam kolom k di posisi diagonal utama.

b. Metode Eliminasi Gauss-Jordan

Bila pada eliminasi Gauss persamaan dasar diubah menjadi matriks triangulasi atas dengan me-nol-kan unsur matriks segitiga bawah [A], maka cara eliminasi Gauss-Jordan dilakukan pula pada bagian segitiga atas matriks. Pada akhir eliminasi, yang tinggal hanyalah suku-suku pada diagonal matriks saja. Bentuk akhir matriks gabungan setelah eliminasi dinyatakan sebagai :

$$\begin{bmatrix} A_{11} & 0 & 0 & \dots & 0 \\ 0 & A_{22} & 0 & \dots & 0 \\ 0 & 0 & A_{33} & \dots & 0 \\ . & . & . & & . \\ 0 & 0 & 0 & \dots & A_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ . \\ B_n \end{Bmatrix}$$

secara umum prosedur me-nol-kan unsur pada posisi atas dan bawah diagonal dilakukan dengan pengali $m_{ik} = \frac{a_{ik}}{a_{kk}}$ bagi unsur di bawah pivotal dan $m_{ik} = \frac{a_{ik}}{a_{kk}}$ bagi unsur di atas pivotal dengan i = 1,2,3,...,n dan I = k-1, k-2,...,1, dan

$A_{ij} = a_{ij\ baru} = a_{ij} - m_{ik}.a_{kj}$, j = 1,2,3,...,n
$B_i = b_{i\ baru} = b_i - m_{ik}.b_k$, untuk i = k+1, k+2, ..., n dan k = 1,2,3,...,n-1
$A_{ik} = a_{ik\ baru} = a_{ik} - m_{ik}.a_{ik}$

Dengan hanya unsur diagonal matriks ≠ 0 dapat dilakukan normalisasi pada matriks. Hasil perhitungan langsung didapatkan pada kolom terakhir matriks. Bentuk matriks gabungan setelah normalisasi adalah sebagai berikut :

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ . & . & . & & . \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{Bmatrix} = \begin{Bmatrix} B_1/A_{11} \\ B_2/A_{22} \\ B_3/A_{33} \\ . \\ B_n/A_{nn} \end{Bmatrix}$$

terlihat dari persamaan di atas bahwa nilai x_i sebagai bilangan yang dicari dapat langsung ditetapkan.

c. Metode Matriks Invers

Eliminasi Gauss-Jordan dapat digunakan untuk mencari invers suatu matriks. Andaikan sebuah persamaan linear simultan

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ . & . & . & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ b_n \end{Bmatrix}$$ dinyatakan dalam notasi matriks sebagai

[A]{x} = {b}, dengan [A] merupakan matriks bujur sangkar dengan syarat unsur-unsur diagonal sudah merupakan nilai poros (pivot). Apabila sama-sama diberlakukan perkalian awal pada matriks [A] dan pada vektor {b} dengan matriks [G], yaitu :

$$[G][A]\{x\} = [G]\{b\}$$

maka dengan memilih $[G][A]^{-1}$, yaitu invers matriks [A], berarti $[G][A] = [A]^{-1}[A] = [I]$, berupa matriks identitas. Penyelesaian [A]{x} = {b} dengan penetuan matriks invers ini dapat dinyatakan sebagai :

$$\{x\} = [A]^{-1}\{b\}$$

prosedur eliminasi Gauss-Jordan yang mengubah matriks [A] menjadi matriks identitas [I] setelah normalisasi menunjukkan bahwa

$$[G][I] = [A]^{-1}$$

untuk memperoleh matriks invers $[A]^{-1}$, unsur matriks [A] disandingkan dengan unsur matriks [I] dalam larik :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & 1 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & 0 & 1 & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & 0 & 0 & 1 & \dots & 0 \\ . & . & . & \dots & \dots & . & . & . & \dots & . \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

dengan prosedur eliminasi Gauss-Jordan, unsur matriks ini diubah menjadi :

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & a^{-1}{}_{11} & a^{-1}{}_{12} & a^{-1}{}_{13} & \dots & a^{-1}{}_{1n} \\ 0 & 1 & 0 & \dots & 0 & a^{-1}{}_{21} & a^{-1}{}_{22} & a^{-1}{}_{23} & \dots & a^{-1}{}_{2n} \\ 0 & 0 & 1 & \dots & 0 & a^{-1}{}_{31} & a^{-1}{}_{32} & a^{-1}{}_{33} & \dots & a^{-1}{}_{3n} \\ . & . & . & \dots & \dots & . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & 1 & a^{-1}{}_{n1} & a^{-1}{}_{n2} & a^{-1}{}_{n3} & \dots & a^{-1}{}_{nn} \end{bmatrix}$$

unsur elemen matriks

$$\begin{bmatrix} a^{-1}_{11} & a^{-1}_{12} & a^{-1}_{13} & \dots & a^{-1}_{1n} \\ a^{-1}_{21} & a^{-1}_{22} & a^{-1}_{23} & \dots & a^{-1}_{2n} \\ a^{-1}_{31} & a^{-1}_{32} & a^{-1}_{33} & \dots & a^{-1}_{3n} \\ . & . & . & \dots & . \\ a^{-1}_{n1} & a^{-1}_{n2} & a^{-1}_{n3} & \dots & a^{-1}_{nn} \end{bmatrix}$$ adalah unsur invers matrik $[A]^{-1}$.

Untuk membuktikan proses eliminasi Gauss-Jordan sebagai cara mencari matriks invers, proses perhitungan pada contoh berikut ini dapat dipelajari.
Tentukan matriks invers dari persamaan :

$$\begin{bmatrix} 5 & -2 & -1 \\ -2.5 & -11.5 & 3.3 \\ 3.25 & -4.5 & 7.5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$ atau [A]{x} = {b}

unsur matrik [A] ditulis dengan unsur matriks [I] di dalam satu larik

$$\begin{bmatrix} 5 & -2 & -1 & 1 & 0 & 0 \\ -2.5 & -11.5 & 3.3 & 0 & 1 & 0 \\ 3.25 & -4.5 & 7.5 & 0 & 0 & 1 \end{bmatrix}$$

proses me-nol-kan unsur di bawah diagonal dapat dilakukan sebagai berikut : baris kedua dikurangi dengan baris pertama yang telah dikalikan dengan (-2.5/5), dan baris ketiga dikurangi dengan baris pertama yang telah dikalikan dengan (2/11.5), dan ini menghasilkan :

$$\begin{bmatrix} 5 & -2 & -1 & 1 & 0 & 0 \\ 0 & -12.5 & 2.80 & 0.5 & 1 & 0 \\ 0 & -3.20 & 8.15 & -0.778 & -0.256 & 1 \end{bmatrix}$$

Baris ketiga dikurangi dengan baris kedua yang telah dikalkan dengan (3.2/12.5), dan ini menghasilkan :

$$\begin{bmatrix} 5 & -2 & -1 & 1 & 0 & 0 \\ 0 & -12.5 & 2.8 & 0.5 & 1 & 0 \\ 0 & 0 & 7.4332 & -0.778 & -0.256 & 1 \end{bmatrix}$$

Baris pertama dikurangi dengan baris kedua yang telah dikalkan dengan (2/12.5), dan ini menghasilkan :

$$\begin{bmatrix} 5 & 0 & -1.448 & 0.98 & -0.16 & 0 \\ 0 & -12.5 & 2.8 & 0.5 & 1 & 0 \\ 0 & 0 & 7.4332 & -0.778 & -0.256 & 1 \end{bmatrix}$$

Baris kedua dikurangi dengan baris ketiga yang telah dikalikan dengan (-2.8/7.4332), baris pertama dengan baris ketiga yang telah dikalikan dengan (-1.448/7.4332), dan ini menghasilkan :

$$\begin{bmatrix} 5 & 0 & 0 & 0.768444 & -0.209987 & 0.194802 \\ 0 & -12.5 & 0 & 0.793064 & 1.096432 & -0.37669 \\ 0 & 0 & 7.4332 & -0.778 & -0.256 & 1 \end{bmatrix}$$

unsur diagonal dijadikan satu satuan

$$\begin{bmatrix} 1 & 0 & 0 & 0.153689 & -0.04197 & 0.03896 \\ 0 & 1 & 0 & -0.06345 & -0.08771 & 0.030135 \\ 0 & 0 & 1 & -0.10467 & -0.03444 & 0.134532 \end{bmatrix}$$

jika bagian diagonal sudah dinormalisasi, maka unsur :

$$\begin{bmatrix} 0.153689 & -0.04197 & 0.03896 \\ -0.06345 & -0.08771 & 0.030135 \\ -0.10467 & -0.03444 & 0.134532 \end{bmatrix} = [A]^{-1}$$

yang dapat dibuktikan dengan perkalian $[A][A]^{-1} = [I]$

$$\begin{bmatrix} 5 & -2 & -1 \\ -2.5 & -11.5 & 3.3 \\ 3.25 & -4.5 & 7.5 \end{bmatrix} \begin{bmatrix} 0.153689 & -0.04197 & 0.03896 \\ -0.06345 & -0.08771 & 0.030135 \\ -0.10467 & -0.03444 & 0.134532 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Penyelesaian persamaan linear simultan dengan menggunakan matriks invers ini adalah :

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{bmatrix} 0.153689 & -0.04197 & 0.03896 \\ -0.06345 & -0.08771 & 0.030135 \\ -0.10467 & -0.03444 & 0.134532 \end{bmatrix} \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$

d. Metode Dekomposisi LU

Dari pembuktian matematika, jika suatu matriks [A] bukanlah singular sifatnya (ada penyelesaian yang unik) :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ . & . & . & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

maka matriks dapat diuraikan sebagai perkalian dua matriks triangular [L] dan [U]. [L] disebut matriks triangular bawah yang elemen matriksnya mempunyai nilai satu pada diagonal dan nilai 0 di atas diagonal seperti :

$$[L] = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ L_{12} & 1 & 0 & \dots & 0 \\ L_{13} & L_{23} & 1 & \dots & 0 \\ . & . & . & \dots & \dots \\ L_{1n} & L_{2n} & L_{3n} & \dots & 1 \end{bmatrix} \text{ dan } [U] = \begin{bmatrix} U_{11} & U_{12} & U_{13} & \dots & U_{1n} \\ 0 & U_{22} & U_{23} & \dots & U_{2n} \\ 0 & 0 & U_{33} & \dots & U_{3n} \\ . & . & . & \dots & \dots \\ 0 & 0 & 0 & \dots & U_{nn} \end{bmatrix}$$

[U] disebut sebagai matriks triangulasi atas dengan nilai elemen di bawah diagonal sama dengan 0.

Dengan demikian :

$$[A] = [L][U]$$

bila persamaan linear simultan dinyatakan dalam matriks $[A]\{x\} = \{b\}$, maka mengisikan matriks [A] dengan [L][U] menghasilkan :

$$[L][U]\{x\} = \{b\}$$

berarti terdapat dua sistem :

$[L]\{z\} = \{b\}$ untuk mencari $\{z\}$, dan $[U]\{x\} = \{z\}$ untuk memperoleh $\{x\}$.

Matriks [U] sama dengan matriks triangulasi yang diperoleh dari metode Gauss. Penyelesaian [U]{x} = {z} dilakukan dengan cara substitusi ke belakang, setelah diketahui nilai {z}, yang diselesaikan dari [L]{z} = {b}.
Unsur elemen matriks [L] merupakan pengali dalam proses eliminasi Gauss, sehingga menyimpan pengali ini selama proses eliminasi menjadi dasar pembentukan matriks [L] dan [U]. Metode penyelesaian seperti ini disebut metode dekomposisi LU.
Sebagai contoh, ditinjau proses dekomposisi LU untuk menyelesaikan persamaan :

$$3x_1 + 2x_2 - 5x_3 = 8$$
$$5x_1 - 2x_2 + 3x_3 = 5$$
$$x_1 + 4x_2 - 2x_3 = 9$$

dalam bentuk matrik : $\begin{bmatrix} 3 & 2 & -5 \\ 5 & -2 & 3 \\ 1 & 4 & -2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 8 \\ 5 \\ 9 \end{Bmatrix}$

dari persamaan awal, terlihat perlu dilakukan pivotisasi untuk koefisien x_1. matriks di atas kemudian diubah susunannya menjadi :

$$\begin{bmatrix} 5 & -2 & 3 \\ 1 & 4 & -2 \\ 3 & 2 & -5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 5 \\ 9 \\ 8 \end{Bmatrix}$$

Proses pertama ialah menghilangkan elemen di bawah a_{11}, menjadi nol. Secara umum, pengali diperoleh dari $m_{ik} = \frac{a_{ik}}{a_{kk}}$.

$\left.\begin{matrix} l = 2 \\ k = 1 \end{matrix}\right\} m_{21} = \frac{a_{21}}{a_{11}} = \frac{1}{5}$ (pengali harus disimpan sebagai unsur elemen l_{21} dari [L])

$a_{21\ baru} = a_{21} - (1/5)a_{11} = 1 - (1/5)(5) = 1 - 1 = 0$
$a_{22\ baru} = a_{22} - (1/5)a_{12} = 4 - (1/5)(-2) = 4 + 0.4 = 4.4$
$a_{23\ baru} = a_{23} - (1/5)a_{13} = -2 - (1/5)(3) = -2 - 0.6 = -2.6$

$\left.\begin{matrix} l = 3 \\ k = 1 \end{matrix}\right\} m_{31} = \frac{a_{31}}{a_{11}} = \frac{3}{5}$ (pengali yang harus disimpan sebagai l_{31})

$a_{31\ baru} = a_{31} - (3/5)a_{11} = 3 - (3/5)(5) = 3 - 3 = 0$
$a_{32\ baru} = a_{32} - (3/5)a_{12} = 2 - (3/5)(-2) = 2 + 1.2 = 3.2$
$a_{33\ baru} = a_{33} - (3/5)a_{13} = -5 - (3/5)(3) = -5 - 1.8 = -6.8$

susunan baru [A] :

$$\begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 3.2 & -6.8 \end{bmatrix}$$

Berdasarkan susunan di atas, tidak ada perubahan pivotal untuk meneruskan proses triangulasi.

$\left.\begin{matrix} i = 3 \\ k = 2 \end{matrix}\right\} m_{32} = \frac{a_{32}}{a_{22}} = \frac{3.2}{4.4} = \frac{8}{11}$ (pengali yang disimpan sebagai l_{32})

$a_{32\ baru} = a_{32} - (8/11)a_{22} = 3.2 - (8/11)(4.4) = 0$
$a_{33\ baru} = a_{33} - (8/11)a_{23} = -6.8 - (8/11)(-2.6) = -4.9091$

sehingga :

$$\begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 3.2 & -6.8 \end{bmatrix} \text{ menjadi } \begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 0 & -4.9091 \end{bmatrix} = [U]$$

sedangkan [L] adalah :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.7273 & 1 \end{bmatrix} = [L]$$

Dekomposisi [A] = [L][U]

$$\begin{bmatrix} 5 & -2 & 3 \\ 1 & 4 & -2 \\ 3 & 2 & -5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.7273 & 1 \end{bmatrix} \begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 0 & -4.9091 \end{bmatrix}$$

Penyelesaian dari persamaan menjadi :

(a) [L]{z} = {b}

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.7273 & 1 \end{bmatrix} \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \end{Bmatrix} = \begin{Bmatrix} 5 \\ 9 \\ 8 \end{Bmatrix} \text{ memberikan } \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \end{Bmatrix} = \begin{Bmatrix} 8 \\ 7.4 \\ -5.382 \end{Bmatrix}$$

(b) [L]{x} = {z}

$$\begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 0 & -4.9091 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 8 \\ 7.4 \\ -5.382 \end{Bmatrix} \text{ memberikan } \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0.180 \\ -0.99 \\ 1.096 \end{Bmatrix}$$

Dari langkah (a), vektor {b} dapat mempunyai nilai yang berbeda, sehingga vektor {z} sebagai vektor antara mendapatkan nilai vektor {x} menjadi fasilitator penyelesaian persamaan [A]{x} = {b} bagi berbagai nilai vektor {b}.

Metode Dekomposisi LU banyak dipakai dalam pemrograman solusi analisis sistem yang baku, yang unsur matriks [A] tetap, tetapi unsur vektor {b} yang terkait dengan pengaru luar terhadap sistem memiliki beberapa variasi.

e. Determinan

Determinan dapat ditentukan dari hasil dekomposisi. Mengingat aturan operasional dua matriks :

$$| [B][C] | = | B ||C|$$

|A| adalah hasil perkalian elemen diagonal matriks [A] bila matriks [A] adalah matriks segitiga atas atau matriks segitiga bawah.

Jika dekomposisi dilakukan tanpa proses pivoting, maka menurut aturan di atas, determinan dapat ditentukan sebagai berikut :

$$|A| = | [L][U} | = | L || U | = | U |$$

Determinan [L] adalah 1, sebab semua unsur elemen diagonal matriks bernilai sama dengan 1. Determinan matriks [U] adalah hasil kali semua elemen diagonalnya. Dengan demikian :

$$|A| = \prod_{i=1}^{n} u_{ii} = \sum a_{11} + a_{22} + \ldots + a_{nn}$$

Jika proses dekomposisi melalui proses pivoting, maka penentuan determinan harus menyertakan matriks permutasi P akibat perubahan susunan baris.

Dua proses terpisah perlu dilakukan, yaitu :

(a) Proses transformasi matriks [A] menjadi matriks $[A^{-1}]$ yaitu matriks dengan unsur elemen diagonal merupakan unsur pivotal dengan [A'] = [P][A] atau [A] = $[P]^{-1}$[A'].
(b) Proses dekomposisi LU matriks [A'] tanpa proses pivoting [A']=[L][U].

Matriks [L] dan [U] terkait dengan matriks [A] sebagai [A] = $[P]^{-1}$[L][U] menurut ketentuan butir (a), sehingga determinan matriks adalah :

$$| A | = | P^{-1} || L || U | = \gamma | U |$$

Nilai γ merupakan nilai $| P^{-1} |$ yang sama dengan -1 atau +1, bergantung pada jumlah ganjil atau genapnya proses pivoting yang dilakukan.

Tinjau penyelesaian determinan bagi matriks :

$$\begin{bmatrix} 3 & 2 & -5 \\ 5 & -2 & 3 \\ 1 & 4 & -2 \end{bmatrix}$$ yang diubah susunannya menjadi $$\begin{bmatrix} 5 & -2 & 3 \\ 1 & 4 & -2 \\ 3 & 2 & -5 \end{bmatrix}$$

melalui proses perkalian.

$$\begin{bmatrix} 5 & -2 & 3 \\ 1 & 4 & -2 \\ 3 & 2 & -5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 2 & -5 \\ 5 & -2 & 3 \\ 1 & 4 & -2 \end{bmatrix}$$

yang menghasilkan matriks triangulasi atas setelah proses eliminasi sebagai :

$$\begin{bmatrix} 5 & -2 & 3 \\ 0 & 4.4 & -2.6 \\ 0 & 0 & -4.9091 \end{bmatrix}$$

determinan | A | = γ | U | = (+1)[5*4.4*-4.9091] = 108.00 mengingat

$$\gamma = \left|P^{-1}\right| = \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix} = +1$$

2. Metode Cramer

Suatu cara yang sederhana dalam menyelesaikan persamaan linear simultan dengan determinan disebut cara Cramer, yang mengambil nama penemunya di tahun 1750, Gabriel Cramer. Untuk menjelaskan aturan Cramer ini, ditinjau sistem persamaan linear simultan

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$

nyatakan determinan dengan unsur-unsur elemen matriks sebagai

$$\begin{vmatrix} (a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & a_{12} & a_{13} \\ (a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & a_{22} & a_{23} \\ (a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & a_{32} & a_{33} \end{vmatrix}$$

mengingat kembali persamaan pertama, maka :

$$\begin{vmatrix} (a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & a_{12} & a_{13} \\ (a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & a_{22} & a_{23} \\ (a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}$$

dengan menggunkan dalil penjumlahan determinan, determinan pada ruas sebelah kiri tanda sama dengan dapat diuraikan sebagai :

$$\begin{vmatrix} a_{11}x_1 & a_{12} & a_{13} \\ a_{21}x_1 & a_{22} & a_{23} \\ a_{31}x_1 & a_{32} & a_{33} \end{vmatrix} + \begin{vmatrix} a_{12}x_2 & a_{12} & a_{13} \\ a_{22}x_2 & a_{22} & a_{23} \\ a_{32}x_2 & a_{32} & a_{33} \end{vmatrix} + \begin{vmatrix} a_{13}x_3 & a_{12} & a_{13} \\ a_{23}x_3 & a_{22} & a_{23} \\ a_{33}x_3 & a_{32} & a_{33} \end{vmatrix}$$

dengan mengeluakan variabel x_i dari determinan dan mengganti persamaan di sebelah kiri tanda sama dengan dengan uraian determinan ini, maka :

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} x_1 + \begin{vmatrix} a_{12} & a_{12} & a_{13} \\ a_{22} & a_{22} & a_{23} \\ a_{32} & a_{32} & a_{33} \end{vmatrix} x_2 + \begin{vmatrix} a_{13} & a_{12} & a_{13} \\ a_{23} & a_{22} & a_{23} \\ a_{33} & a_{32} & a_{33} \end{vmatrix} x_3 = \begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}$$

dengan memperhatikan suku kedua dan suku ketiga determinan yang mempunyai unsur identik antara kolom pertama dan kedua, dan kolom pertama dan ketiga, maka nilai kedua determinan ini = 0, sehingga :

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} x_1 = \begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}$$

menghasilkan :

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

dengan prosedur serupa, x_2 dapat diperoleh dengan menyatakan determinan dengan unsur elemen matriks sebagai :

$$\begin{vmatrix} a_{11} & (a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & a_{13} \\ a_{21} & (a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & a_{23} \\ a_{31} & (a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & a_{33} \end{vmatrix}$$

mengingat persamaan : $\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$, maka :

$$\begin{vmatrix} a_{11} & (a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & a_{13} \\ a_{21} & (a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & a_{23} \\ a_{31} & (a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}$$

atau

$$\begin{vmatrix} a_{11} & a_{11}x_1 & a_{13} \\ a_{21} & a_{21}x_1 & a_{23} \\ a_{31} & a_{31}x_1 & a_{33} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{12}x_2 & a_{13} \\ a_{21} & a_{22}x_2 & a_{23} \\ a_{31} & a_{32}x_2 & a_{33} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{13}x_3 & a_{13} \\ a_{21} & a_{23}x_3 & a_{23} \\ a_{31} & a_{33}x_3 & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}$$

sehingga :

$$\begin{vmatrix} a_{11} & a_{11} & a_{13} \\ a_{21} & a_{21} & a_{23} \\ a_{31} & a_{31} & a_{33} \end{vmatrix} x_1 + \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} x_2 + \begin{vmatrix} a_{11} & a_{13} & a_{13} \\ a_{21} & a_{23} & a_{23} \\ a_{31} & a_{33} & a_{33} \end{vmatrix} x_3 = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}$$

dengan memperhatikan suku pertama dan suku ketiga determinan yang mempunyai unsur identik antara kolom pertama dan kolom kedua, dan kolom kedua dan ketiga, maka nilai kedua determinan ini = 0, sehingga :

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} x_2 = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}$$

menghasilkan : $$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

Untuk mencari x_3, ditetapkan bentuk determinan :

$$\begin{vmatrix} a_{11} & a_{12} & (a_{11}x_1 + a_{12}x_2 + a_{13}x_3) \\ a_{21} & a_{22} & (a_{21}x_1 + a_{22}x_2 + a_{23}x_3) \\ a_{31} & a_{32} & (a_{31}x_1 + a_{32}x_2 + a_{33}x_3) \end{vmatrix}$$

seperti prosedur untuk mendapatkan x1 dan x2, diperoleh :

$$x_3 = \frac{\begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

Berdasarkan penjelasan di atas, secara umum aturan Cramer dalam penyelesaian persamaan linear simultan adalah dengan menetapkan determinan dari variabel | A | dan kemudian penentuan variabel dilakukan dari pembagian $x_i = \frac{|A_i|}{|A|}$ dengan | A_i | merupakan determinan matriks dengan menukarkan unsur matriks [A] kolom ke-i dengan vektor {b}.

3. Metode Iterasi

Pada metode solusi persamaan linear simultan yang telah dibahas, proses hitungan eliminasi berupa pengurangan suatu baris dengan baris lainnya dapat dilakukan dengan terpenuhinya syarat perilaku koefisien variabel – dalam susunan persamaan poros. Apabila koefisien dari variabel antara satu baris dengan yang lainya hampir sama, pengurangan dua baris akan menghasilkan nilai yang sangat kecil. Walaupun ujud koefisien dinyatakan dalam nilai eksak, hasil proses tidak akan cukup cermat. Metode iterasi merupakan cara proses perhitungan yang dapat melakukan koreksi kesalahan selagi pemrosesan dilaksanakan, sehingga kekurang-cermatan hasi seperti cara eliminasi dapat dihindarkan.

a. Iterasi Jacobi

Iterasi Jacobi menggunakan rumusan rekursif untuk menghitung nilai pendekatan solusi persamaan. Proses iterasi dilakukan sampai dicapai suatu nilai yang konvergen dengan toleransi yang diberikan. Untuk menjelaskan proses iterasi, tinjaulah suatu persamaan linear simultan derajat tiga :

$$a_{11}X_1 + a_{12}X_2 + a_{13}X_3 = b_1$$
$$a_{21}X_1 + a_{22}X_2 + a_{23}X_3 = b_2$$
$$a_{31}X_1 + a_{32}X_2 + a_{33}X_3 = b_3$$

Persamaan ini dapat dinyatakan dalam bentuk sebagai berikut :

$$x_1 = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2 - a_{13}x_3\right) = F_1\left(x_1, x_2, x_3\right)$$

$$x_2 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 - a_{23}x_3\right) = F_2\left(x_1, x_2, x_3\right)$$

$$x_3 = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1 - a_{32}x_2\right) = F_3\left(x_1, x_2, x_3\right)$$

dengan syarat nilai a_{11}, a_{22}, dan $a_{33} \neq 0$, apabila ditetapkan nilai awal x_1, x_2, dan x_3 sebagai $x_1^0 = x_1$; $x_2^0 = x_2$; $x_3^0 = x_3$, maka untuk mendapatkan pendekatan pertama dilakukan proses :

$$x_1^{(1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)}\right) = F_1\left(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\right)$$

$$x_2^{(1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(0)} - a_{23}x_3^{(0)}\right) = F_2\left(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\right)$$

$$x_3^{(1)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(0)} - a_{32}x_2^{(0)}\right) = F_3\left(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\right)$$

Pendekatan kedua dengan nilai $x_1^{(1)} = x_1$; $x_2^{(1)} = x_2$; $x_3^{(1)} = x_3$, mendapatkan koreksi perhitungan dari iterasi :

$$x_1^{(2)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(1)} - a_{13}x_3^{(1)}\right) = F_1\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\right)$$

$$x_2^{(2)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(1)}\right) = F_2\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\right)$$

$$x_3^{(2)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)}\right) = F_3\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\right)$$

Proses iterasi dilanjutkan dengan menggunakan pendekatan ke (I-1) untuk mendapatkan nilai x_1, x_2, x_3 bagi pendekatan ke-i. Konvergensi hasil diperoleh jika hasil pendekatan memberikan nilai konstan atau simpangan antara dua hasil lebih kecil dari toleransi kesalahan yang ditetapkan. Secara umum, apabila akan dicari variabel n-persamaan dengan proses iterasi, maka prosedur yang telah diuraikan dapat digunakan. Untuk iterasi ke-i, perhitungan secara umum dinyatakan sebagai :

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)}\right)$$

...

$$x_3^{(k+1)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} - \ldots - a_{1n}x_n^{(k)}\right)$$

Penetapan nilai variabel menurut proses ini disebut iterasi Jacobi. Dengan nilai awal sembarang $x_i^{(0)}$, ada kemungkinan konvergensi tercapai secara lambat, sehingga perlu ditetapkan syarat terjadinya konvergensi dalam perhitungan iterasi, yaitu :

Maksimum $\left(\frac{1}{|a_{ii}|}\sum_{j\neq i} a_{ij}\right) < 1$ (i = 1,2,...,n; j=1,2,3,...,n).

b. Iterasi Gauss-Seidel

Iterasi Gauss-Seidel sebagai cara penyelesaian persamaan linear simultan tidak jauh berbeda dengan iterasi Jacobi. Pada iterasi Gauss-Seidel, nilai hasil perhitungan pada baris awal langsung digunakan untuk perhitungan nilai selanjutnya di dalam iterasi. Dengan metode ini konvergensi akan tercapai lebih cepat. Bentuk iteratif persamaan linear adalah sebagai berikut :

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)}\right)$$

...

$$x_3^{(k+1)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} - \ldots - a_{1n}x_n^{(k)}\right)$$

Prosedur perhitungan dan syarat konvergensi iterasi sama seperti pada iterasi Jacobi.

4. Metode Cholesky

Dalam ilmu rekayasa, persamaan linear simultan yang diperoleh dari rumusan matematika berdasarkan teori elastis umumnya mempunyai unsur koefisien variabel yang simetris. Persamaan linear simultan tersebut dapat dinyatakan dengan :

$$\begin{aligned}
A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + \ldots + A_{1n}X_n &= B_1\\
A_{21}X_1 + A_{22}X_2 + A_{23}X_3 + \ldots + A_{2n}X_n &= B_2\\
A_{31}X_1 + A_{32}X_2 + A_{33}X_3 + \ldots + A_{3n}X_n &= B_3\\
&\ldots\\
A_{n1}X_1 + A_{n2}X_2 + A_{n3}X_3 + \ldots + A_{nn}X_n &= B_n
\end{aligned}$$

Atau dalam notasi matrik :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n}\\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n}\\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n}\\ . & . & . & \ldots & \ldots\\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn}\end{bmatrix}\begin{Bmatrix} x_1\\ x_2\\ x_3\\ .\\ x_n\end{Bmatrix} = \begin{Bmatrix} b_1\\ b_2\\ b_3\\ .\\ b_n\end{Bmatrix} \text{ atau } [A]\{x\} = \{b\}$$

Matriks [A] disebut matriks simetri apabila diluar unsur diagonal, unsur matriks baris sama dengan unsur matriks kolom pada indeks baris dan kolom yang sama. Dengan demikian, unsur matriks simetri dirumuskan sebagai $a_{ij} = a_{ji}$; i≠j; i=1,2,3,...,n; j=1,2,3,...,n.

Matrisk simetri dapat dinyatakan dalam produk matriks triangulasi bawah dengan matriks triangulasi atas, dengan kedua matriks satu sama lain adalah matriks transpose. Faktorisasi matriks [A] dalam kedua matriks adalah : $[A] = [U]^T[U]$.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} u_{11} & 0 & 0 & \dots & 0 \\ u_{12} & u_{22} & 0 & \dots & 0 \\ u_{13} & u_{23} & u_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ u_{1n} & u_{2n} & u_{3n} & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Berdasarkan bentuk matriks di atas, unsur matriks [A] merupakan hasil produk unsur baris dalam matriks $[U]^T$ dan unsur kolom matriks [U]. Hubungan unsur a_{ij} dengan unsur u_{ij} baris pertama :

$a_{11} = u_{11}^2$; $a_{12} = u_{11}u_{12}$; $a_{13} = u_{11}u_{13}$; ... ; $a_{1n} = u_{11}a_{1n}$

Nyatakan u_{1i} dalam unsur a_{1i} :

$$a_{22} = \sqrt{a_{11}}; u_{12} = \frac{a_{12}}{u_{11}} = \frac{a_{12}}{\sqrt{a_{11}}}; u_{13} = \frac{a_{13}}{u_{11}} = \frac{a_{13}}{\sqrt{a_{11}}}; \dots; u_{1n} = \frac{a_{1n}}{u_{11}} = \frac{a_{1n}}{\sqrt{a_{11}}}$$

baris kedua :

$a_{22} = u_{12}^2 + u_{22}^2$; $a_{23} = u_{12}u_{13} + u_{22}u_{23}$; ... ; $a_{2n} = u_{12}u_{1n} + u_{22}u_{2n}$

Nyatakan u_{2i} dalam unsur a_{2i} :

$$u_{22} = \sqrt{a_{22} - u_{12}^2} = \sqrt{a_{22} - \frac{a_{12}^2}{a_{11}}}$$

$$u_{23} = \frac{a_{23} - u_{12}u_{13}}{u_{22}} = \frac{a_{23} - \left(\frac{a_{12}}{\sqrt{a_{11}}}\right)\left(\frac{a_{13}}{\sqrt{a_{11}}}\right)}{\sqrt{a_{22} - \frac{a_{12}^2}{a_{11}}}}$$

.

.

$$u_{2n} = \frac{a_{2n} - u_{12}u_{1n}}{u_{22}} = \frac{a_{2n} - \left(\frac{a_{12}}{\sqrt{a_{11}}}\right)\left(\frac{a_{1n}}{\sqrt{a_{11}}}\right)}{\sqrt{a_{22} - \frac{a_{12}^2}{a_{11}}}}$$

Baris ketiga :

$a_{33} = u_{13}^2 + u_{23}^2 + u_{33}^2$; ... ; $a_{3n} = u_{13}u_{1n} + u_{23}u_{2n} + u_{33}u_{3n}$

Nyatakan unsur u_{3i} dalam unsur a_{3i} :

$$u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2} = \sqrt{a_{33} - \frac{a_{13}^2}{a_{11}} - \left[\frac{a_{23} - \left(\frac{a_{12}}{\sqrt{a_{11}}}\right)\left(\frac{a_{13}}{\sqrt{a_{11}}}\right)}{\sqrt{a_{22} - \frac{a_{12}^2}{a_{11}}}}\right]^2}$$

...

$u_{3n} = \frac{a_{3n} - u_{13}u_{1n} - u_{23}u_{2n}}{u_{33}}$, dengan nilai u_{ij} didapat dari perhitungan sebelumnya.

Rumusan umum untuk menyatakan unsur matriks [A] pada posisi diagonal :

$a_{ii} = u_{1i}^2 + u_{2i}^2 + u_{3i}^2 + \ldots + a_{ii}^2$ atau $a_{ii} = \sum_{k=1}^{i} u_{ki}^2 (i = j)$

unsur matriks di luar posisi diagonal :

$a_{ij} = u_{1i}\,u_{1j} + u_{2i}\,u_{2j} + u_{3i}\,u_{3j} + \ldots + u_{ii}\,u_{ij}$ atau $a_{ij} = \sum_{k=1}^{i} u_{ki} u_{kj} (i < j)$

sehingga rumusan umum untuk menyatakan unsur matriks [U] adalah :

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2 A} \qquad (1 < i = j)$$

$$u_{ij} = \frac{1}{u_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj} \right) \quad (1 < i < j)$$

$u_{ij} = 0 \qquad (i > j)$

tiga persamaan di atas disebut formula faktorisasi yang mengubah unsur matriks [A] menjadi unsur dari dua matriks $[U]^T$ dan [U]. cara ini dinamakan metode Akar Cholesky karena adanya unsur akar pada pernyataan u_{ii}, dan hanya berlaku bagi matriks yang simetri serta nilai di bawah tanda akar adalah bilangan positif. Untuk menjelaskan metode ini, tinjau matriks simetri berikut ini :

$$[A] = \begin{bmatrix} 9 & -3 & 6 \\ -3 & 17 & -10 \\ 6 & -10 & 12 \end{bmatrix}$$

untuk mendapatkan [U] = $\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$ dengan menggunakan tiga persamaan yang telah didefinisikan sebelumnya, diperoleh :

$$u_{11} = \sqrt{9} = 3; u_{12} = \frac{a_{12}}{\sqrt{a_{11}}} = \frac{-3}{3} = -1; u_{13} = \frac{a_{13}}{\sqrt{a_{11}}} = \frac{6}{3} = 2$$

$$u_{22} = \sqrt{a_{22} - u_{12}^2} = \sqrt{17 - 1} = 4$$

$$u_{23} = \frac{a_{23} - u_{12} u_{13}}{u_{22}} = \frac{-10 + 2}{4} = -2$$

$$u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2} = \sqrt{12 - 4 - 4} = 2$$

sehingga :

$$[U] = \begin{bmatrix} 3 & -1 & 2 \\ 0 & 4 & -2 \\ 0 & 0 & 2 \end{bmatrix}; [U]^T = \begin{bmatrix} 3 & 0 & 0 \\ -1 & 4 & 0 \\ 2 & -2 & 2 \end{bmatrix}$$

dapat dibuktikan bahwa :

$$[U]^T [U] = \begin{bmatrix} 3 & -1 & 2 \\ 0 & 4 & -2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ -1 & 4 & 0 \\ 2 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 9 & -3 & 6 \\ -3 & 17 & -10 \\ 6 & -10 & 12 \end{bmatrix}$$

Apabila matriks simetri tidak memenuhi nilai positif difinitif, faktorisasi dilakukan ke dalam produk :

$$[A]=[\overline{U}]^T[D][\overline{U}]$$

Matriks [D] merupakan matriks diagonal dengan unsur-unsur matrik berupa suku kuadrat dari faktorisasi baris matriks [U]. Jika suku terfaktor adalah u_{ii}, maka unsur diagonal dalam matriks [D] adalah : $d_{ii} = u_{ii}^2$; i = 1,2,3,...,n

Nilai kuadrat uii menghindarkan perhitungan di bawah tanda akar. Dengan keterangan ini, perlu dilakukan modifikasi, dimana modifikasi ini yang dinamakan dengan Cara Cholesky.

Persamaan $[A]=[\overline{U}]^T[D][\overline{U}]$ secara lengkap dapat diberikan sebagai berikut :

$$[A]=\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \overline{u}_{12} & 1 & 0 & \dots & 0 \\ \overline{u}_{13} & \overline{u}_{23} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \overline{u}_{1n} & \overline{u}_{2n} & \overline{u}_{3n} & \dots & 1 \end{bmatrix}\begin{bmatrix} d_{11} & 0 & 0 & \dots & 0 \\ 0 & d_{22} & 0 & \dots & 0 \\ 0 & 0 & d_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & d_{nn} \end{bmatrix}\begin{bmatrix} 1 & \overline{u}_{12} & \overline{u}_{13} & \dots & \overline{u}_{1n} \\ 0 & 1 & \overline{u}_{23} & \dots & \overline{u}_{2n} \\ 0 & 0 & 1 & \dots & \overline{u}_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Unsur matriks [A] di posisi diagonal berdasarkan rumusan di atas adalah $a_{11} = d_{11}$, pada suku pertama. Suku pada posisi diagonal lainnya adalah :

$$a_{ii} = d_{ii}\overline{u}_{1i}^2 + d_{22}\overline{u}_{2i}^2 + d_{33}\overline{u}_{3i}^2 + \dots + d_{ii}$$

Unsur matriks [A] di luar posisi diagonal dinyatakan sebagai :

$$a_{ij} = d_{11}\overline{u}_{1i}\overline{u}_{1j} + d_{22}\overline{u}_{2i}\overline{u}_{2j} + d_{33}\overline{u}_{3i}\overline{u}_{3j} + \dots + d_{ii}\overline{u}_{ij} = d_{ii}u_{ij} + \sum_{k=1}^{(i-1)} d_{kk}\overline{u}_{ki}\overline{u}_{kj}$$

nilai (1 < I < j)

unsur matriks [D] dan matriks [U] dapat diperoleh dengan menggunakan persamaan pertama dan ke dua di atas dalam persamaan faktorisasi.

$d_{11} = a_{11}$

$$d_{22} = a_{22} - d_{11}\overline{u}_{11}^2$$

$$d_{33} = a_{33} - d_{11}\overline{u}_{13}^2 - d_{22}\overline{u}_{23}^2$$

...

$$d_{nn} = a_{nn} - d_{11}\overline{u}_{1n}^2 - d_{22}\overline{u}_{2n}^2 - \dots - d_{n-1,n-1}\overline{u}_{n-1,n}^2$$

atau secara umum :

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} d_{kk}\overline{u}_{ki}^2 \; ; \; (1 < i = j)$$

Cara mendapatkan unsur matriks [U] pada persamaan faktorisasi di atas adalah :

$$\overline{u}_{ii} = 1; \overline{u}_{12} = \frac{a_{12}}{d_{11}}; \overline{u}_{13} = \frac{a_{13}}{d_{11}}; \dots; \overline{u}_{1n} = \frac{a_{1n}}{d_{11}}$$

$$\overline{u}_{23} = \frac{1}{d_{22}}\left(a_{23} - d_{11}\overline{u}_{12}\overline{u}_{13}\right); \overline{u}_{24} = \frac{1}{d_{22}}\left(a_{24} - d_{11}\overline{u}_{12}\overline{u}_{14}\right); \dots; \overline{u}_{2n} = \frac{1}{d_{22}}\left(a_{2n} - d_{11}\overline{u}_{12}\overline{u}_{1n}\right)$$

atau secara umum :

$$\overline{u}_{ij} = \frac{1}{d_{ii}}\left(a_{ij} - \sum_{k=1}^{i-1} d_{kk}\overline{u}_{ki}\overline{u}_{kj}\right); 1 < i < j; \overline{u}_{ij} = 0, i > j$$

Jika diuraikan dalam bentuk matriks pada contoh :

$$[A]=\begin{bmatrix} 9 & -3 & 6 \\ -3 & 17 & -10 \\ 6 & -10 & 12 \end{bmatrix}$$, untuk mendapatkan $$[D]=\begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix}$$

dan $[\overline{U}]=\begin{bmatrix}1 & \overline{u}_{12} & \overline{u}_{13}\\ 0 & 1 & \overline{u}_{23}\\ 0 & 0 & 1\end{bmatrix}$ dengan menggunakan kedua persamaan umum yang telah didefinisikan di atas akan diperoleh :

$[D]=\begin{bmatrix}9 & 0 & 0\\ 0 & 16 & 0\\ 0 & 0 & 4\end{bmatrix}$, dan $[\overline{U}]=\begin{bmatrix}1 & -\frac{1}{3} & \frac{2}{3}\\ 0 & 1 & -\frac{1}{2}\\ 0 & 0 & 1\end{bmatrix}$

dapat dibuktikan bahwa $[\overline{U}]^T[D][\overline{U}]=[A]$

Urutan penyelesaian adalah mendapatkan suku pada diagonal d_{ii}, kemudian menghitung unsur pada baris ke-i dari matriks [U]. Perhitungan ini dapat diubah dalam urutan kolom, sehingga :

$$\overline{u}_{ij}=\frac{1}{d_{ii}}\left(a_{ij}-\sum_{k=1}^{i-1}d_{kk}\overline{u}_{ki}\overline{u}_{kj}\right);1<i<j \qquad d_{jj}=a_{jj}-\sum_{k=1}^{j-1}d_{kk}\overline{u}_{kj}^2\ ;\ (1<i=j)$$

perkalian antara $d_{kk}\times\overline{u}_{kj}$ terdapat pada kedua persamaan di atas, tetapan $\overline{u}^*_{kj}=d_{kk}\times\overline{u}_{kj}$ dan hitung unsur u_{ij}, d_{ij} dengan rumusan :

$$\overline{u}^*_{ij}=a_{ij}-\sum_{k=1}^{i-1}\overline{u}_{ki}\overline{u}^*_{kj}\ ;\ (1<i<j)$$

$$d_{jj}=a_{jj}-\sum_{k=1}^{j-1}\overline{u}_{kj}\overline{u}^*_{kj}\ ;\ (1<i=j),\ \text{dengan}\ \overline{u}_{ki}=\frac{1}{d_{kk}}\overline{u}^*_{kj}$$

nilai $\overline{u}^*_{ij}$ yang merupakan nilai antara di kolom-j diperoleh untuk setiap unsur di luar diagonal setelah kolom pertama. Unsur diagonal djj dihitung dari persamaan $d_{jj}=a_{jj}-\sum_{k=1}^{j-1}d_{kk}\overline{u}_{kj}^2$, yang pada saat bersamaan nilai akhir dari unsur $\overline{u}_{kj}$ diproses dengan persamaan $\overline{u}_{ki}=\frac{1}{d_{kk}}\overline{u}^*_{kj}$.

Dengan urutan perhitungan ini, jumlah perkalian dapat dikurangi dibanding dengan cara Akar Cholesky, dan perhitungan akar dihindari. Apabila persamaan linear simultan yang akan diselesaikan sebagai [A]{x} = {b}, dengan menggati matriks $[A]=[\overline{U}]^T[D][\overline{U}]$, berarti bahwa persamaan linear dapat dinyatakan sebagai : $[\overline{U}]^T[D][\overline{U}]\{x\}=\{b\}$.

Namakan : $[\overline{U}]\{x\}=\{y\}$, yaitu : $\begin{bmatrix}1 & \overline{u}_{12} & \overline{u}_{13} & \dots & \overline{u}_{1n}\\ 0 & 1 & \overline{u}_{23} & \dots & \overline{u}_{2n}\\ 0 & 0 & 1 & \dots & \overline{u}_{3n}\\ \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & \dots & 1\end{bmatrix}\begin{Bmatrix}x_1\\ x_2\\ x_3\\ \dots\\ x_n\end{Bmatrix}=\begin{Bmatrix}y_1\\ y_2\\ y_3\\ \dots\\ y_n\end{Bmatrix}$

Dan

$[D]\{y\}=\{z\}$, yaitu : $\begin{bmatrix} d_{11} & 0 & 0 & \dots & 0 \\ 0 & d_{22} & 0 & \dots & 0 \\ 0 & 0 & d_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & d_{nn} \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_n \end{Bmatrix} = \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \\ \dots \\ z_n \end{Bmatrix}$

maka, dengan vektor {b} yang diketahui unsurnya, dapat ditetapkan nilai antara vektor {z} dari persamaan

$[\overline{U}]^T\{z\}=\{b\}$, yaitu $\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \overline{u}_{12} & 1 & 0 & \dots & 0 \\ \overline{u}_{13} & \overline{u}_{23} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \overline{u}_{1n} & \overline{u}_{2n} & \overline{u}_{3n} & \dots & 1 \end{bmatrix} \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \\ \dots \\ z_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{Bmatrix}$

dengan substitusi ke depan : $z_1 = b_1$

$z_2 = b_2 - \overline{u}_{12} z_1$

$z_3 = b_3 - \overline{u}_{13} z_1 - \overline{u}_{23} z_2$, secara umum :

$$z_i = b_i - \sum_{k=1}^{i-1} \overline{u}_{ki} z_k$$

dengan mengetahui vektor {z}, dari persamaan $[D]\{y\}=\{z\}$ diperoleh unsur vektor {y}, yaitu

$y_i = \frac{z_i}{d_{ii}}$; dengan i = 1,2,3,...,n

proses perhitungan terakhir adalah menemukan vektor {x} dari persamaan $[\overline{U}]\{x\}=\{y\}$ dengan cara substitusi ke belakang.

$x_n = y_n$

$x_{n-1} = y_{n-1} - \overline{u}_{n-1,n} x_n$

$$x_i = y_i - \sum_{k=i+1}^{n} \overline{u}_{ik} x_k ; (i < n)$$

catatan :
Seluruh metode penyelesaian persamaan simultan linear dapat anda temukan pada buku-buku yang membahas masalah Metode Numerik.

Solusi Persamaan Linear Simultan Metode Gauss

```
Program Gauss;

{
program Eliminasi Gauss

Daftar variabel yang digunakan :
       A[], B[] : matrix pembentuk persamaan linear simultan
       G[]      : matrix gabungan
       x[]      : matrix hasil perhitungan
}

Uses WinCrt;

Type
    Mat55 = Array[0..5,0..5] of real;
    mat56 = Array[0..5,0..6] of real;

var
   A             : Mat55;
   G             : Mat56;
   b,x           : Array[0..5] of real;
   factor,dummy  : real;
   i,j,k,n       : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     b[0]:=16;
     b[1]:=10;
     b[2]:=12;
     n:=3;
end;

begin
     inisialisasi;
     { membentuk matrix gabungan }
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               G[i][j]:=A[i][j];
          end;
     end;
     for j:=0 to n-1 do
     begin
          G[j][n]:=b[j];
     end;
     writeln(' Matrix penggabungan ');
     for i:=0 to n-1 do
     begin
          for j:=0 to n do
          begin
               write(G[i][j]:11:3);
```

```
            end;
            writeln;
        end;
        { proses eliminasi }
        for i:=0 to n-2 do
        begin
            for j:=i+1 to n-1 do
            begin
                factor:=G[j][i]/G[i][i];
                for k:=i to n do
                begin
                    G[j][k]:=G[j][k]-factor*G[i][k];
                end;
            end;
        end;
        writeln(' Matrix setelah dieliminasi ');
        for i:=0 to n-1 do
        begin
            for j:=0 to n do
            begin
                write(G[i][j]:11:3);
            end;
            writeln;
        end;
        { substitusi mundur }
        x[n-1]:=G[n-1][n]/G[n-1][n-1];
        for k:=0 to n-2 do
        begin
            i:=n-2-k;
            dummy:=G[i][n];
            for j:=i+1 to n-1 do
            begin
                dummy:=dummy-G[i][j]*x[j];
            end;
            x[i]:=dummy/G[i][i];
        end;
        writeln(' Hasil akhir ');
        for i:=0 to n-1 do
        begin
            writeln('x[',i:2,'] = ',x[i]:10:3);
        end;
end.
```

Solusi Persamaan Linear Simultan Metode Gauss-Jordan

```
program Gauss_Jordan;

{
  Program Eliminasi dengan pivoting Gauss-Jordan

  Daftar variabel yang digunakan :
       A[], B[] : matrix pembentuk persamaan linear simultan
       G[]      : matrix gabungan
       x[]      : matrix hasil perhitungan
}

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5] of real;
    Mat56       = Array[0..5,0..6] of real;
```

```
Var
   A            : Mat55;
   G            : Mat56;
   b            : Array[0..5] of real;
   factor,dummy : real;
   i,j,k,n      : integer;

procedure inisialisasi;
begin
     A[0][0]:=1;
     A[0][1]:=3;
     A[0][2]:=1;
     A[1][0]:=1;
     A[1][1]:=2;
     A[1][2]:=5;
     A[2][0]:=4;
     A[2][1]:=1;
     A[2][2]:=2;
     b[0]:=10;
     b[1]:=12;
     b[2]:=16;
     n:=3;
end;

begin
     inisialisasi;
     { bentuk matrix gabungan }
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               G[i][j]:=A[i][j];
          end;
     end;
     for j:=0 to n-1 do
     begin
          G[j][n]:=b[j];
     end;
     writeln(' Matrix penggabungan ');
     for i:=0 to n-1 do
     begin
          for j:=0 to n do
          begin
               write(G[i][j]:10:2);
          end;
          writeln;
     end;
     {  proses eliminasi  }
     for i:=0 to n-1 do
     begin
          { proses pivoting }
          dummy:=abs(G[i][i]);
          k:=i;
          for j:=i+1 to n-1 do
              if (abs(G[j][i])>dummy) then
              begin
                   dummy:=abs(G[j][i]);
                   k:=j;
              end;
              if (k<>i) then
              begin
                 for j:=i to n do
                 begin
```

```
                    dummy:=G[k][j];
                    G[k][j]:=G[i][j];
                    G[i][j]:=dummy;
                end;
            end;
        for j:=0 to n-1 do
        begin
            if (i<>j) then
            begin
                factor:=G[j][i]/G[i][i];
                for k:=i to n do G[j][k]:=G[j][k]-factor*G[i][k];
            end;
        end;
    end;
    for i:=0 to n-1 do
    begin
        G[i][n]:=G[i][n]/G[i][i];
        G[i][i]:=1;
    end;
    writeln(' matrix setelah eliminasi ');
    for i:=0 to n-1 do
    begin
        for j:=0 to n do
        begin
            write(G[i][j]:10:2);
        end;
        writeln;
    end;
    writeln(' Hasil akhir ');
    for i:=0 to n-1 do
    begin
        writeln('x[',i:2,'] = ',G[i][n]:10:2);
    end;
end.
```

Solusi Persamaan Linear Simultan Metode Inversi Matrix

```
program Invers_Matrix;

{ program solusi persamaan linear simultan dengan metode Inversi Matrix
digabung dengan Eliminasi Gauss-Jordan (pivoting)

  Daftar Variabel yang digunakan :
  A[]             : Matrix segiempat
  G[]             : Matrix gabungan
  x[]             : Matrix hasil perhitungan }

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5] of real;
    Mat510      = Array[0..5,0..10] of real;

var
   A            : Mat55;
   G            : Mat510;
   factor,dummy : real;
   i,j,k,n      : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
```

```
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     n:=3;
end;

begin
     inisialisasi;
     { bentuk matrix gabungan }
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               G[i][j]:=A[i][j];
          end;
     end;
     for j:=0 to n-1 do
     begin
          G[j][j+n]:=1;
     end;
     writeln(' Matrix gabungan ');
     for i:=0 to n-1 do
     begin
          for j:=0 to 2*n-1 do
          begin
               write(G[i][j]:10:2);
          end;
          writeln;
     end;
     { proses eliminasi  }
     for i:=0 to n-1 do
     begin
          { pivoting process }
          dummy:=abs(G[i][i]);
          k:=i;
          for j:=i+1 to n-1 do
              if (abs(G[j][i])>dummy) then
              begin
                   dummy:=abs(G[j][i]);
                   k:=j;
              end;
          if (k<>i) then
             for j:=i to 2*n-1 do
             begin
                  dummy:=G[k][j];
                  G[k][j]:=G[i][j];
                  G[i][j]:=dummy;
             end;
          for j:=0 to n-1 do
              if (i<>j) then
              begin
                   factor:=G[j][i]/G[i][i];
                   for k:=i to 2*n-1 do
                   begin
                        G[j][k]:=G[j][k]-factor*G[i][k];
                   end;
              end;
     end;
     for i:=0 to n-1 do
```

```
      begin
           for j:=n to 2*n-1 do
           begin
                G[i][j]:=G[i][j]/G[i][i];
           end;
           G[i][i]:=1;
      end;
      writeln(' Matrix setelah eliminasi ');
      for i:=0 to n-1 do
      begin
           for j:=0 to 2*n-1 do
           begin
                write(G[i][j]:10:5);
           end;
           writeln;
      end;
      writeln(' Hasil invers matrix ');
      for i:=0 to n-1 do
      begin
           for j:=n to 2*n-1 do
           begin
                write(G[i][j]:10:5);
           end;
           writeln;
      end;
      { pemeriksaan [A] x invers [A] }
      for i:=0 to n-1 do
      begin
           for k:=0 to n-1 do
           begin
                G[i][k]:=0;
                for j:=0 to n-1 do
                begin
                     G[i][k]:=G[i][k]+A[i][j]*G[j][n+k];
                end;
           end;
      end;
      writeln(' perkalian [A] dengan invers [A] ');
      for i:=0 to n-1 do
      begin
           for j:=0 to n-1 do
           begin
                write(G[i][j]:10:5);
           end;
           writeln;
      end;
end.
```

Solusi Persamaan Linear Simultan dengan Metode Dekomposisi LU

```
program Dekomposisi_Matrix;

{ program solusi persamaan linear simultan dengan metode Dekomposisi Matrix

  Daftar Variabel yang digunakan :
  A[],b[]          : Matrix pembentuk persamaan linear simultan
  L[],U[]          : Matrix dekomposisi
  x[]              : Matrix hasil perhitungan }

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5] of real;
```

```
var
   A,L,U        : Mat55;
   factor,dummy : real;
   i,j,k,n      : integer;
   b,x          : array[0..5] of real;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     b[0]:=16;
     b[1]:=10;
     b[2]:=12;
     n:=3;
end;

begin
     inisialisasi;
     { dekomposisi matrix }
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               U[i][j]:=A[i][j];
          end;
     end;
     for i:=0 to n-1 do
     begin
          L[i][i]:=1;
     end;
     for k:=0 to n-2 do
     begin
          for j:=k+1 to n-1 do
          begin
               L[j][k]:=U[j][k]/U[k][k];
               for i:=k to n-1 do
               begin
                    U[j][i]:=U[j][i]-L[j][k]*U[k][i];
               end;
          end;
     end;
     writeln(' Matrix [A]');
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               write(A[i][j]:11:7);
          end;
          writeln;
     end;
     writeln(' Matrix [L] : ');
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
```

```
                write(L[i][j]:11:7);
            end;
            writeln;
        end;
        writeln(' Matrix [U] :');
        for i:=0 to n-1 do
        begin
            for j:=0 to n-1 do
            begin
                write(U[i][j]:11:7);
            end;
            writeln;
        end;
        { substitusi ke depan }
        for i:=1 to n-1 do
        begin
            for j:=0 to i-1 do
            begin
                b[i]:=b[i]-L[i][j]*b[j];
            end;
        end;
        { substitusi ke belakang }
        b[n-1]:=b[n-1]/U[n-1][n-1];
        for k:=0 to n-2 do
        begin
            i:=n-2-k;
            for j:=i+1 to n-1 do
            begin
                b[i]:=b[i]-U[i][j]*b[j];
            end;
            b[i]:=b[i]/U[i][i];
        end;
        writeln(' Hasil akhir : ');
        for i:=0 to n-1 do
        begin
            writeln('x[',i:2,'] = ',b[i]:11:7);
        end;
end.
```

Solusi Persamaan Linear Simultan dengan metode Determinan

```
program determinan;

{ solusi persamaan linear simultan dengan metode determinan matrix

  daftar variabel yang digunakan :
  A[]              : matrix segiempat
  L[],U[]          : matrix dekomposisi
  Det              : determinan matrix          }

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5]of real;

Var
   A,L,U        : Mat55;
   Det          : real;
   i,j,k,n      : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
```

```
     A[0][1]:=1;
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     n:=3;
end;

begin
     inisialisasi;
     { dekomposisi matrix }
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               U[i][j]:=A[i][j];
          end;
     end;
     for i:=0 to n-1 do
     begin
          L[i][i]:=1;
     end;
     for k:=0 to n-2 do
     begin
          for j:=k+1 to n-1 do
          begin
               L[j][k]:=U[j][k]/U[k][k];
               for i:=k to n-1 do
               begin
                    U[j][i]:=U[j][i]-L[j][k]*U[k][i];
               end;
          end;
     end;
     writeln(' Matrix [A]');
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               write(A[i][j]:10:7);
          end;
          writeln;
     end;
     writeln(' Matrix [L]');
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               write(L[i][j]:10:7);
          end;
          writeln;
     end;
     writeln(' Matrix [U]');
     for i:=0 to n-1 do
     begin
          for j:=0 to n-1 do
          begin
               write(U[i][j]:10:7);
          end;
          writeln;
     end;
```

```
     Det:=1;
     for i:=0 to n-1 do
     begin
          Det:=Det*U[i][i];
     end;
     writeln(' Determinan matrix = ',Det:10:3);
end.
```

Solusi persamaan linear simultan dengan metode Cramer

```
Program Cramer;
{ solusi persamaan linear simultan dengan metode cramer

  daftar variabel yang digunakan :
  A[], b[]              : matrix pembentuk persamaan linear simultan
  ADJ[]                 : matrix penyimpanan sementara untuk perhitungan
determinan
  L[],U[]               : dekomposisi matrix
  Det_A                 : determinan matrix A
  x[]                   : matrix hasil perhitungan }

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5] of real;

Var
   A,ADJ        : Mat55;
   b,x          : array[0..5] of real;
   Det_A        : real;
   i,j,k,n      : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     b[0]:=16;
     b[1]:=10;
     b[2]:=12;
     n:=3;
end;

function Determinan(U:Mat55):real;
var
   L    : Mat55;
   dummy: real;
   i,j,k: integer;

begin
     { dekomposisi matrix }
     for i:=0 to n-1 do L[i][i]:=1.0;
     for k:=0 to n-2 do
     begin
          for j:=k+1 to n-1 do
          begin
               L[j][k]:=U[j][k]/U[k][k];
```

```
                for i:=k to n-1 do U[j][i]:=U[j][i]-L[j][i]*U[k][i];
            end;
        end;
        dummy:=1;
        for i:=0 to n-1 do dummy:=dummy*U[i][i];
        Determinan:=dummy;
end;

begin
        inisialisasi;
        { memindahkan matrix A ke ADJ }
        for i:=0 to n-1 do
        begin
            for j:=0 to n-1 do
            begin
                ADJ[i][j]:=A[i][j];
            end;
        end;
        Det_A:=Determinan(ADJ);
        for i:=0 to n-1 do
        begin
            { membentuk matrix adjoin A }
            for j:=0 to n-1 do
            begin
                if (i=j) then for k:=0 to n-1 do ADJ[k][j]:=b[k]
                else for k:=0 to n-1 do ADJ[k][j]:=A[k][j];
            end;
            x[i]:=Determinan(ADJ)/Det_A;
        end;
        writeln(' Hasil Akhir ');
        for i:=0 to n-1 do
        begin
            writeln('x[',i:2,'] = ',x[i]:10:6);
        end;
end.
```

Solusi persamaan linear dengan metode Iterasi Jacobi

```
Program Jacobi;

{ solusi persamaan linear simultan dengan metode iterasi Jacobi

  Daftar variabel yang digunakan :
  A[],b[]          : matrix pembentuk persamaan linear simultan
  x[]              : matrix hasil perhitungan
  Eps              : toleransi yang diperbolehkan        }

Uses WinCrt;

Type
    Mat55       = Array [0..5,0..5] of real;

Var
   A,ADJ                : Mat55;
   b,x,x_1              : Array[0..5] of real;
   Eps                  : real;
   i,j,n,True,iter      : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
     A[0][2]:=2;
```

```
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     b[0]:=16;
     b[1]:=10;
     b[2]:=12;
     n:=3;
     Eps:=0.0000001;
end;

begin
     inisialisasi;
     True:=0;
     iter:=0;
     { inisialisasi }
     for i:=0 to n-1 do
     begin
          x[i]:=0;
     end;
     { melakukan iterasi sampai konvergen }
     while (True=0) do
     begin
          iter:=iter+1;
          True:=1; { asumsi telah konvergen }
          for i:=0 to n-1 do
          begin
               x_1[i]:=b[i];
               for j:=0 to n-1 do
               begin
                    if(i<>j) then x_1[i]:=x_1[i]-A[i][j]*x[j];
               end;
               x_1[i]:=x_1[i]/A[i][i];
               if (abs(x_1[i]-x[i])>Eps) then True:=0; { belum konvergen }
          end;
          { menuliskan hasil iterasi }
          write(' Iterasi ke ',iter:3,' : ');
          for i:=0 to n-1 do
          begin
               write(x_1[i]:10:7);
          end;
          writeln;
          for i:=0 to n-1 do
          begin
               x[i]:=x_1[i];
          end;
     end;
     writeln(' Hasil akhir : ');
     for i:=0 to n-1 do
     begin
          writeln('x[',i:2,'] = ',x[i]:10:6);
     end;
end.
```

Solusi persamaan linear simultan dengan metode iterasi Gauss-Seidel

```
program Gauss_Seidel;

{ program solusi persamaan linear iterasi Gauss-Seidel
```

```
  Daftar Variabel yang digunakan :
  A[],b[]              : matrix pembentuk persamaan linear simultan
  x[]                  : matrix hasil perhitungan
  Eps                  : toleransi yang diperbolehkan }

Uses WinCrt;

Type
    Mat55       = Array[0..5,0..5] of real;
    Mat56       = Array[0..5,0..6] of real;

Var
    A                   : Mat55;
    b,x                 : Array[0..5] of real;
    factor,xb,Eps       : real;
    i,j,n,True,iter     : integer;

procedure inisialisasi;
begin
     A[0][0]:=4;
     A[0][1]:=1;
     A[0][2]:=2;
     A[1][0]:=1;
     A[1][1]:=3;
     A[1][2]:=1;
     A[2][0]:=1;
     A[2][1]:=2;
     A[2][2]:=5;
     b[0]:=16;
     b[1]:=10;
     b[2]:=12;
     n:=3;
     Eps:=0.0000001;
end;

begin
     inisialisasi;
     true:=0;
     iter:=0;
     { inisialisasi }
     for i:=0 to n-1 do x[i]:=0;
     { melakukan iterasi sampai hasilnya konvergen }
     while (true=0) do
     begin
          iter:=iter+1;
          true:=1; { asumsi telah konvergen }
          write(' Iterasi ke ',iter:2,' : ');
          for i:=0 to n-1 do
          begin
               xb:=b[i];
               for j:=0 to n-1 do
                   if (i<>j) then xb:=xb-A[i][j]*x[j];
               xb:=xb/A[i][i];
               if (Abs(xb-x[i])>Eps) then true:=0; { belum konvergen }
               x[i]:=xb;
               write(xb:11:7);
          end;
          writeln;
     end;
     writeln(' Hasil akhir : ');
     for i:=0 to n-1 do
     begin
          writeln('x[',i:2,'] = ',x[i]:11:7);
```

```
    end;
end.
```

Solusi persamaan linear simultan dengan metode Cholesky

```
program Cholesky;

{ solusi persamaan linear simultan dengan metode Cholesky }

Uses WinCrt;

Const       Max=5;
Type        MM    = Array[1..max,1..max] of real;
            M     = Array[1..max] of real;
Var         n     : integer;
            a     : MM;
            x     : M;

procedure factor;
var
   j,j1,i,i1,k  : integer;
   sum,temp     : real;

begin
     if(a[1][1]<=0.0) then writeln(' Warning!! non-positive stiffness dof 1');
     for j:=2 to n do
     begin
          j1:=j-1;
          if (j1<>1) then
             for i:=2 to j1 do
             begin
                  sum:=a[i][j];
                  i1:=i-1;
                  for k:=1 to i1 do sum:=sum-a[k][i]*a[k][j];
                  a[i][j]:=sum;
             end;
          sum:=a[j][j];
          for k:=1 to j1 do
          begin
               temp:=a[k][j]/a[k][k];
               sum:=sum-temp*a[k][j];
               a[k][j]:=temp;
          end;
          if (sum<=0) then writeln(' Warning!!! non-positive stiffness dof',j);
          a[j][j]:=sum;
     end;
end;

procedure solver;
var
   i,k1,k,i1,k2 : integer;
   sum          : real;

begin
     for i:=1 to n do
     begin
          sum:=x[i];
          k1:=i-1;
          if (i<>1) then
             for k:=1 to k1 do sum:=sum-a[k][i]*x[k];
          x[i]:=sum;
     end;
     for i:=1 to n do x[i]:=x[i]/a[i][i];
```

```
     for i1:=1 to n do
     begin
          i:=n-i1+1;
          sum:=x[i];
          k2:=i+1;
          if (i<>n) then
             for k:=k2 to n do sum:=sum-a[i][k]*x[k];
          x[i]:=sum;
     end;
end;

procedure init;
begin
     n:=4;
     a[1][1]:=76800.0;
     a[2][2]:=336534.0;
     a[2][4]:=-200000.0;
     a[3][3]:=270870.0;
     a[4][4]:=373358.0;
     x[2]:=-54.0;
end;

procedure output;
var i:integer;
begin
     writeln;
     for i:=1 to n do writeln('x[',i,'] = ',x[i]:11:9);
     writeln;
end;

begin
     init;
     factor;
     solver;
     output;
end.
```