

ILK1106

DASAR PEMROGRAMAN

Disusun oleh:

1. Dian Rachmawati, S.Si, M.Kom
2. Stephen J. Rusli
3. Aurick Daffa Muhammad
4. Andreas M. Lumban Gaol
5. Frederick Godiva
6. Harry Hamara
7. Alvian

No. Dokumen	: IK-GKM-Fasilkom-TI-010-DP
Berlaku Efektif	: 18 September 2023
Revisi	: 10
Direvisi	: TIM GKM ILK Fasilkom-TI
Diperiksa/Disetujui	: GJM dan GKM ILK Fasilkom-TI USU
Disahkan Oleh	: Dr. Maya Silvi Lydia B.Sc., M.Sc. Dekan Fasilkom-TI USU



**S-1 ILMU KOMPUTER
FASILKOM-TI
USU**

<p>MODUL PRAKTIKUM</p>	No. Dokumen	:	IK-GKM-ILK-FASILKOM-TI-010-DP
	Edisi	:	01
	Revisi	:	10
	Berlaku Efektif	:	18 September 2023
	Halaman	:	ii dari iii
	KATA PENGANTAR		

Puji syukur kehadirat Allah SWT. Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya sehingga penyusun dapat menyelesaikan buku penuntun praktikum ini. Sholawat dan salam kepada Nabi Muhammad saw. sosok manusia teladan bagi hidup dan kehidupan manusia sampai akhir zaman.

Modul praktikum ini disusun berdasarkan silabus mata kuliah praktikum Program Studi S1 Ilmu Komputer Fasilkom-TI USU. Buku ini terdiri atas beberapa modul yang terdiri dari tujuan, teori, latihan dan tugas praktikum dengan harapan mahasiswa dapat membuat jurnal sebagai tolak ukur bagi keberhasilan pencapaian tujuan praktikum. Buku penuntun praktikum ini merupakan buku pedoman dasar dan bahan ajar dalam pelaksanaan praktikum.

Atas tersusunnya buku ini, penyusun mengucapkan terima kasih kepada Ketua dan Sekretaris Program Studi S1 Ilmu Komputer, Bapak/Ibu Dosen, Staf Tata Usaha Program Studi S1 Ilmu Komputer dan semua pihak yang telah membantu dalam penyusunan buku penuntun praktikum ini. Buku penuntun praktikum digunakan oleh Program Studi S1 Ilmu Komputer Fasilkom-TI USU dan merupakan implementasi dari pelaksanaan Sistem Manajemen Mutu Perguruan Tinggi.

Buku penuntun ini adalah revisi dari buku penuntun praktikum terdahulu. Penyusun menyadari buku ini masih jauh dari kesempurnaan oleh karena itu penyusun sangat mengharapkan kritik dan saran dari para pembaca demi menyempurnakan buku penuntun praktikum ini. Dengan adanya buku penuntun praktikum diharapkan dapat membantu mahasiswa dalam pelaksanaan praktikum dan bermanfaat secara maksimal bagi semua pihak.

Medan, 08 September 2023
 Program Studi S1 Ilmu Komputer
 Fasilkom-TI USU
 Ketua,

Dr. Amalia, ST., M.T.
 NIP. 19781221 201404 2 001

<p style="text-align: center;">MODUL PRAKTIKUM</p>	No. Dokumen	:	IK-GKM-ILK-FASILKOM-TI-010-DP
	Edisi	:	01
	Revisi	:	10
	Berlaku Efektif	:	18 September 2023
	Halaman	:	iii dari iii
	DAFTAR ISI		

KATA PENGANTAR..... ii

DAFTAR ISI..... iii

MODUL I PENGENALAN PEMROGRAMAN DAN BAHASA

PEMROGRAMAN PASCAL 1

MODUL II OPERATOR 7

MODUL III PENGGUNAAN OPERASI KONDISI..... 12

MODUL IV PERULANGAN DAN PELONCATAN 18

MODUL V ARRAY..... 26

MODUL VI PROSEDUR DAN FUNGSI 32

MODUL VII RECORD..... 37

MODUL VIII POINTER 44

MODUL I

PENGENALAN PEMROGRAMAN DAN BAHASA PEMROGRAMAN PASCAL

I. Tujuan

1. Praktikan dapat memahami apa itu Algoritma, Program, dan Bahasa Pemrograman.
2. Praktikan dapat memahami apa itu Bahasa Pemrograman Pascal.
3. Praktikan dapat memahami apa Tipe Data dan Jenis Tipe Data dalam Pascal.

II. Teori

1. Algoritma dan Program

Algoritma adalah serangkaian langkah-langkah terstruktur yang dirancang secara sistematis untuk memecahkan masalah. Konsep kunci dalam algoritma adalah kejelasan logis. Setiap langkah dalam algoritma haruslah tersusun secara logis dan dapat dievaluasi sebagai benar atau salah..

Algoritma tidak selalu identik dengan ilmu komputer saja. Di kehidupan sehari-hari, banyak proses yang dapat dijelaskan menggunakan algoritma. Sebagai contoh, cara membuat nasi kuning atau hidangan dalam suatu resep juga bisa dianggap sebagai algoritma. Setiap resep selalu mengikuti langkah-langkah tertentu dalam urutan tertentu. Jika langkah-langkah tersebut tidak teratur atau tidak logis, hasil masakan yang diinginkan mungkin tidak akan tercapai. Algoritma bisa ditampilkan dalam bentuk bahasa ilmiah, *pseudocode*, atau *flowchart*.

Contoh Algoritma

1. Menghitung luas dan keliling lingkaran
 - Tentukan nilai jari-jari lingkaran
 - Pi merupakan konstanta dengan nilai 3.14
 - Menghitung luas dan keliling
$$L = \pi * r * r$$
$$K = 2 * \pi * r$$
 - Tulis hasil perhitungan luas dan keliling
2. Menghitung rata-rata 3 buah bilangan
 - Tentukan nilai bilangan 1, bilangan 2, dan bilangan 3
 - Jumlahkan ketiga bilangan tersebut
 - Bagi 3 hasil jumlah ketiga bilangan
 - Tulis hasilnya

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Program = Algoritma + Bahasa Pemrograman.

Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa, aturan-aturan tata bahasanya, pernyataan-pernyataannya, tata cara pengoperasian *compiler*-nya dan memanfaatkan pernyataan-pernyataan tersebut untuk membuat program yang ditulis dalam bahasa itu saja.

Tahapan dalam pemrograman

1. Definisikan Masalah

Berikut hal-hal yang harus diketahui dalam menganalisis suatu masalah:

- Kondisi awal, *input* yang tersedia
- Kondisi akhir, *output* yang diinginkan
- Data dan operator yang tersedia
- Syarat dan kendala yang harus dipenuhi

2. Buat Algoritma dan Struktur Cara Penyelesaian

3. Menulis Program Dalam Suatu Bahasa Pemrograman Tertentu

4. Mencari Kesalahan

5. Uji dan Verifikasi Program

6. Dokumentasi Program

7. Pemeliharaan Program

2. Struktur Program Pascal

Bentuk struktur penulisan program dalam bahasa pemrograman Pascal secara sederhana adalah sebagai berikut.

```
program <judul program>
uses <daftar unit>
<bagian deklarasi>
begin
  <statemen-statemen>
end.
```

Gambar1 Struktur penulisan program Pascal

Keterangan:

- a) **<judul program>** biasanya sama dengan nama program, contoh:

```
[-----]
[ program ujicoba; ]
```

- b) **<daftar unit>** berisi daftar unit yang akan digunakan di dalam program. Untuk membatasi unit satu dengan yang lainnya, gunakan tanda koma (,). Contoh:

```
[-----]
[ uses crt; ]
```

- c) **<bagian deklarasi>** berisi macam-macam deklarasi yang dibutuhkan oleh program, yang meliputi hal-hal berikut:

- Deklarasi tipe data
- Deklarasi konstanta
- Deklarasi variabel
- Deklarasi prosedur
- Deklarasi fungsi
- Deklarasi label

- d) <**statemen-statement**> merupakan perintah-perintah yang akan dituliskan. Statement-statement ini harus berada di dalam blok begin-end. Berikut ini contoh penulisan statemen-statement di dalam program.

```
begin
    clrscr;
    writeln('S-1 Ilmu Komputer USU');
end.
```

3. Prosedur Write dan Read

Dalam pembuatan program tentunya akan melakukan banyak penulisan (*output*) dan pembacaan (*input*) data. Dalam bahasa Pascal, prosedur yang digunakan untuk melakukan penulisan adalah **Write**, sedangkan untuk melakukan pembacaan adalah **Read**. Prosedur **Write** dan **Read** kadang kala diakhiri dengan **In** (singkatan dari *line*) menjadi **Writeln** dan **Readln** yang berguna untuk memindahkan kursor ke baris baru di bawahnya setelah program melakukan proses *input* ataupun *output*. Prosedur **Read** juga bisa diakhiri dengan **key** menjadi **Readkey** yang berfungsi untuk membaca sebuah karakter. Prosedur **Readln** dan **Readkey** juga digunakan di akhir program agar hasil dari program dapat dilihat.

4. Komentar Program

Ada dua macam cara untuk membuat komentar program pada bahasa Pascal:

- Menggunakan tanda (* ... *) atau { ... }
- Digunakan untuk menuliskan komentar yang banyaknya lebih dari satu baris.
- Menggunakan tanda // ...
- Digunakan untuk menuliskan satu baris komentar.

NB: Jangan menuliskan sintaks program di sebelah kanan // karena sintaks akan dianggap sebagai komentar

Contoh:

```
begin // ini komentar 1 baris di kanan sintaks program
    clrscr;
    {ini komentar 2 baris, yang berada di dalam kurung kurawal}
    writeln('S-1 Ilmu Komputer USU');
end.
```

5. Variabel

Adalah suatu pengenal yang didefinisikan oleh programmer untuk menyimpan nilai atau data tertentu yang dibutuhkan dalam program pada saat program sedang berjalan (*run time*). Bentuk umum pendeklarasian variabel:

```
var
    NamaVariabel : tipe_data;
var
    x      : integer;
    nama   : string[25];
```

6. Konstanta

"Konstanta adalah variabel yang bernilai tetap selama program berjalan. Bentuk umum konstanta:

```
const
    NamaKonstanta1 : nilaiKonstanta1;
    NamaKonstanta2 : nilaiKonstanta2;
```

7. Tipe Data

a) Bilangan Bulat

Berikut ini tipe data yang termasuk ke dalam tipe bilangan bulat di dalam bahasa Pascal:

Tipe Data	Penjelasan	Alokasi Memori
Shortint	Tipe data dengan nilai bilangan bulat mulai dari -128 hingga 127	(1 Byte)
Integer	Tipe data dengan nilai bilangan bulat mulai dari -32768 hingga 32767	(2 Byte)
Longint	Tipe data dengan nilai bilangan bulat mulai dari -2147483648 hingga 2147383647	(4 Byte)
Byte	Tipe data dengan nilai bilangan bulat mulai dari 0 hingga 255	(1 Byte)
Word	Tipe data dengan nilai bilangan bulat mulai dari 0 hingga 65535	(2 Byte)

b) Bilangan Riil

Adalah bilangan yang mengandung angka di belakang koma. Tipe data yang termasuk ke dalam tipe bilangan riil:

Tipe Data	Rentang (Jangkauan) Nilai
Real	2.9×10^{-39} sampai 1.7×10^{38}
Single	1.5×10^{-45} sampai 3.4×10^{38}
Double	5.0×10^{-324} sampai 1.7×10^{308}
Extended	3.4×10^{-4932} sampai 1.1×10^{4932}
Comp	-9.2×10^{-18} sampai 9.2×10^{18}

c) Karakter

Tipe ini direpresentasikan dengan menggunakan tipe data **char**. Berguna untuk merepresentasikan nilai-nilai karakter seperti 'A', 'B', 'C', 'D', 'E'.

d) String

Tipe ini direpresentasikan dengan menggunakan tipe data **string**. String merupakan kumpulan dari karakter yang terangkai menjadi satu kata ataupun kalimat, misalnya 'ILKOM JAYA'.

e) Tipe Logika

Tipe yang hanya memiliki dua buah nilai yaitu **true** dan **false**. Pascal menyediakan tipe khusus untuk mendefinisikan data-data *logic* tersebut, yaitu tipe data **Boolean**.

8. Penulisan Identifier

Identifier merupakan suatu nama yang digunakan sebagai pengenal dari suatu variabel, fungsi, konstanta, *struct*, objek dan sebagainya. Penulisan *identifier* yang sah harus mengikuti aturan berikut:

1. Gabungan dari huruf – huruf. Contoh: **nama**, **saya**, **aku**, **satu** dan sebagainya.
2. Gabungan dari huruf dan angka dimana harus diawali dengan huruf terlebih dahulu. Contoh : **N01**, **N4ma**, **Say4**, **s4t0** dan sebagainya.
3. Dimulai dengan tanda *underscore*. Contoh: **_aku**, **_kamu**, **_dia**, **_1satu**
4. Apabila lebih dari satu kata digunakan tanda *underscore*. Contoh: **saya_makan**, **aku_cinta_dia**, **fungsi_satu**, **konstanta_1** dan sebagainya.
5. Tidak berupa sintaks/prosedur bawaan dari Pascal. Contoh: **write**, **read**, **for**, **case**, **if**, **dll**.

III. Tugas

1. Buatlah program untuk menampilkan “**Belajar Pascal untuk Dasar Pemrograman**”!
 2. Buatlah program untuk menampilkan daftar nama bulan!
- *Catatan: output diurutkan ke bawah.
3. Buat program untuk menampilkan biodata pribadi Anda, data-data Anda diinputkan terlebih dahulu setelah itu ditampilkan!

Contoh :

Input :
Masukkan Nama Anda : Aurick Muhammad
Masukkan NIM Anda : 211401022
Masukkan Kom Anda : A
Masukkan Alamat Anda : Jalan Bunga Rinte
Output :
Hallo, Nama saya Aurick Muhammad , NIM saya 211401022 , saya di kom A , alamat saya di Jalan Bunga Rinte . Salam Kenal Semua!

- *Catatan :
 - Masing-masing diinput ke dalam satu variabel.
 - Tulisan tebal merupakan inputan.

4. Buat program Pascal untuk menampilkan alamat lengkap Anda!

Contoh :

Input :
Jalan : Alumni
No : 10
Kota : Medan
Kode Pos : 20155
Output :
Alamat : Jalan Alumni No. 10 Medan 20155

- *Catatan :
 - Masing-masing diinput ke dalam satu variabel.
 - Tulisan tebal merupakan inputan.

5. Buatlah sebuah algoritma berdasarkan kehidupan sehari-hari, yang memiliki minimal 10 langkah penyelesaian.

MODUL II OPERATOR

I. Tujuan

- Praktikan memahami operator-operator serta dapat menggunakan operator dalam pemrograman Pascal.

II. Tugas Persiapan

- Apa pengertian operator, operand dan operasi secara bahasa dan istilah?
- Jelaskan mengapa operator diperlukan dalam pemrograman!
- Sebutkan operator-operator yang digunakan dalam kehidupan sehari-hari beserta contoh!
- Sebutkan 2 sifat operator dalam bahasa Pascal serta contoh operator dan penggunaannya!
- Operator apa saja yang terdapat dalam:
 - Flowchart I/O* (jajar genjang)
 - Flowchart Proses* (persegi)
 - Flowchart Kondisi* (belah ketupat)

III. Teori

Tanda operasi (*operator*) dalam Pascal dikelompokkan ke dalam 9 kategori, yaitu: *assignment, binary, unary, bitwise, relational, logical, address, set, and string* operator.

1. Assignment Operator

Assignment operator digunakan untuk memberikan nilai. Operator ini menggunakan simbol titik dua diikuti oleh tanda sama dengan :=

Contoh: A := B;

2. Binary Operator

Operator ini digunakan untuk mengoperasikan dua buah operand. Operand dapat berbentuk konstanta/variabel. Operator ini digunakan untuk operasi aritmatika yang berhubungan dengan nilai tipe data integer dan *floating point*.

Operator	Operasi
*	Perkalian
DIV	Pembagian bulat
/	pembagian real
MOD	Modulus
+	Pertambahan
-	Pengurangan

3. Unary Operator

Operator ini hanya menggunakan satu operand saja. Operand dapat berbentuk konstanta atau variabel. *Unary* operator dapat berupa *unary minus* (-) atau plus (+).

4. Bitwise Operator

Operator ini digunakan untuk operasi bit per bit pada integer. Operasi yang dipergunakan antara lain: **NOT**, **AND**, **OR**, **XOR**, **Shl**, dan **Shr**.

Operator	Operasi
NOT	bitwise negation
AND	bitwise and
OR	bitwise or
XOR	bitwise xor
SHL	shift left
SHR	shift right

5. Relational Operator

Operator ini digunakan untuk membandingkan hubungan antara dua buah operand dan menghasilkan tipe boolean, yaitu *true* atau *false*.

Operator	Operasi
=	sama dengan
<>	tidak sama dengan
>	lebih besar dari
>=	lebih besar sama dengan
<	lebih kecil dari
<=	lebih kecil sama dengan
IN	seleksi dari anggota

6. Logical Operator

Logical operator memiliki empat operasi, yaitu: **not**, **and**, **or**, dan **xor**. Bentuk operator ini sama dengan *bitwise operator*, tetapi penggunaannya lain. Operator ini bekerja dengan nilai-nilai logika *true* dan *false*.

7. Address Operator

Operator ini khusus berhubungan dengan alamat di memori, yaitu *address of operator* (@) dan *indirection operator* (^). Operator @ akan menghasilkan alamat dari suatu nilai variabel dan operator ^ akan memberikan nilai di alamat yang ditunjukkan.

8. Set Operator

Set operator digunakan untuk operasi himpunan.

Operator	Operasi
+	<i>union / sum</i>
-	<i>set difference</i>
*	<i>Intersection</i>

9. String Operator

Operator ini digunakan untuk operasi string. Hanya ada sebuah operator string saja, yaitu operator + yang digunakan untuk menggabungkan string.

IV. Latihan

```
begin
    writeln(17 * 6);
    writeln(24 / 12);
    writeln(26 div 5);
    writeln(26 mod 5);
end.
```

Program 2.1 Penggunaan binary operator

```
begin
    writeln(not 5);
    writeln(9 or 3);
    writeln(not -5);
    writeln(10 and 7);
    writeln(2 xor 4);
    writeln(4 shl 2);
    writeln(120 shr 5);
end.
```

Program 2.2 Penggunaan bitwise operator

```
var
    a,b : integer;
begin
    a := 8;
    b := 6;
    writeln(a = b);
    writeln(a = 8);
    writeln(a <> b);
    writeln(a > b);
    writeln(a < b);
end.
```

Program 2.3 Penggunaan relational operator

```
begin
    writeln(not True);
    writeln(True and False);
    writeln(False or True);
    writeln(True xor False);
end.
```

Program 2.4 Penggunaan logical operator

```
var
    nilai    : integer;
    pointer : ^integer;
begin
    nilai    := 20;
    pointer := @nilai;
    writeln(pointer^);
end.
```

Program 2.5 Penggunaan address operator

```
var
    lab1, lab2, lab3, lab : string[20];
begin
    lab1 := 'Praktikum ';
    lab2 := 'Dasar ';
    lab3 := 'Pemrograman';
    lab := lab1 + lab2 + lab3;
    writeln('Lab : ', lab);
end.
```

Program 2.6 Penggunaan string operator

V. Tugas

- Buatlah program untuk menghitung jumlah gaji yang diterima seorang karyawan PT. Berjaya Abadi dalam sebulan dengan ketentuan sebagai berikut:
 - Input-an* : nama karyawan, gaji pokok banyak hari lembur, banyak hari tidak bekerja.
 - Syarat untuk menghitung jumlah gaji yang diterima dalam sebulan adalah :
 - Gaji lembur = $Rp150.000 \times$ banyak hari lembur
 - Potongan gaji = - $Rp30.000$ setiap 1 hari tidak bekerja
 - Gaji total merupakan jumlah dari Gaji pokok + Gaji lembur – Potongan gaji

Input	Output
Nama Karyawan : Ian Brown Gaji Pokok : 6000000 Lembur (hari) : 4 Tidak Bekerja (hari) : 3	Gaji Total : Rp 6510000

- Buatlah program untuk menghitung nilai akhir seorang mahasiswa dengan ketentuan sebagai berikut:
 - Inputan : nama mahasiswa, nilai keaktifan, nilai tugas dan nilai ujian.
 - Proses yang dilakukan untuk mendapatkan nilai murni dari masing-masing nilai adalah:
 - Nilai murni keaktifan = Nilai keaktifan $\times 20\%$
 - Nilai murni tugas = Nilai tugas $\times 35\%$
 - Nilai murni ujian = Nilai ujian $\times 45\%$
 - Nilai akhir adalah Nilai murni keaktifan + Nilai murni tugas + Nilai murni ujian.

Input	Output
Nama Mahasiswa : Putri Nilai Keaktifan : 85 Nilai Tugas : 90 Nilai Ujian : 95	Nilai Akhir : 91.25

- Buatlah program untuk menghitung luas dan keliling sebuah lapangan sepak bola

Input	Output
Panjang : 5 Meter Lebar : 2 Meter	Luas lapangan : 10 Meter Keliling lapangan : 14 Meter

- Buatlah sebuah program dari beberapa pilihan program di bawah ini:
 - Menghitung luas dan keliling sebuah segitiga dengan sisi bangun datar diketahui.
 - Mencari nilai Sinus, Cosinus, dan Tangen dengan sudut sebagai *input-an*.
 - Mengubah derajat temperatur dari Celsius (*input-an*) ke derajat Fahrenheit, Kelvin dan Reamur.
- Tulislah perhitungan manual untuk penggunaan Bitwise Operator: or, and, xor, shl, dan shr.
**Catatan: -Nilai bilangan ditentukan oleh masing-masing praktikan.*

MODUL III

PENGGUNAAN OPERASI KONDISI

I. Tujuan

1. Praktikan dapat memahami penggunaan operasi kondisi dan operasi logika dalam pembuatan program kondisi.
2. Praktikan dapat mengaplikasikan Operasi Kondisi dalam sebuah program.

II. Tugas Persiapan

1. Apa pengertian dari operator kondisi?
2. Operasi kondisi apa saja yang terdapat dalam Pascal?
3. Buatlah *flowchart* untuk melakukan pengecekan *input*-an nilai x lebih kecil dari 5!
4. Berikanlah contoh operasi kondisi dalam kehidupan sehari-hari!

III. Teori

Operasi kondisi adalah proses menjalankan suatu perintah pada program dengan *output* yang berbeda sesuai dengan hasil dari kondisi-kondisi. *Output* akan ditampilkan apabila kondisi yang ditetapkan terpenuhi (*true*), atau akan menghasilkan *output* yang berbeda saat kondisi tidak terpenuhi (*false*) atau langsung keluar dari blok operasi.

1. Statement If

Statement if akan diikuti oleh kondisi yang akan diperiksa, dan disusul oleh kata kunci **then**. Setelah kata kunci **then**, tulis perintah yang akan dijalankan jika hasil kondisi yang diperiksa sebelumnya bernilai 1 (*true*).

1.a Statement if Satu Kasus

Statement if satu kasus berarti perintah akan dijalankan hanya jika kondisi yang diperiksa sebelumnya bernilai benar. Apabila kondisi yang diperiksa bernilai salah, maka perintah tidak akan dijalankan.

Bentuk umumnya adalah:

```
if (kondisi yang diperiksa) then
    perintah yang dijalankan;
```

Latihan 3.1 Statement if Satu Kasus (v.1)

```
program cek_Bilangan_Lebih_Besar_Dari_10;
uses crt;
var
  x : integer;

begin
  clrscr;
  write ('Masukkan sebuah bilangan bulat: '); readln(x);

  if (x > 10) then
    writeln(x, ' > 10');

  readln;
end.
```

Latihan 3.2 Statement if Satu Kasus (v.2)

```

program cek_Huruf_Vokal;
uses crt;
var
    huruf : char;

begin
    clrscr;
    write('Masukkan sebuah huruf: '); readln(huruf);
    writeln;

    if ((huruf = 'a') or (huruf = 'A') or
        (huruf = 'e') or (huruf = 'E') or
        (huruf = 'i') or (huruf = 'I') or
        (huruf = 'o') or (huruf = 'O') or
        (huruf = 'u') or (huruf = 'U')) then

        writeln('(',huruf, ') merupakan huruf vokal');

    readln;
end.

```

1.b Statement if Dua Kasus

Jika di bentuk sebelumnya *output* hanya akan ditampilkan jika kondisi yang diperiksa bernilai benar, di *statement if* dua kasus, program akan memeriksa kondisi dan akan menampilkan *output* apabila hasil pemeriksaan kondisi bernilai benar atau salah.

Bentuk umumnya adalah:

```

if (kondisi) then
    perintah yang dijalankan jika kondisi terpenuhi/TRUE
else
    perintah yang dijalankan jika kondisi tidak terpenuhi/FALSE;

```

NB: Bentuk di atas berlaku jika statemen di setiap kondisi HANYA satu baris saja, jika lebih dari satu baris maka:

```

if (kondisi) then
    begin
        perintah_1;
        ...
        perintah_x;
    end
else
    begin
        perintah_y;
    end;

```

NB: Perlu dingat, jika ada perintah sebelum else, perintah end tersebut TIDAK BOLEH diakhiri dengan titik koma (;). Misalnya pada bentuk umum pertama, sebelum else (...terpenuhi/TRUE) tidak ada tanda titik koma (;), sama juga pada bentuk umum yang kedua, perintah end di dalam if tidak ada titik koma (;) karena program masih dilanjutkan oleh else.

Latihan 3.3 Statement if Dua Kasus (v.1)

```
program cek_Genap_Ganjil;
uses crt;
var
    bil : integer;

begin
    clrscr;
    write('Masukkan bilangan bulat: '); readln(bil);

    if (bil mod 2 = 0) then
        writeln(bil,' adalah bilangan GENAP')
    else
        writeln(bil,' adalah bilangan GANJIL');

    readln;
end.
```

Latihan 3.4 Statement if Dua Kasus (v.2)

```
program cek_Positif_Negatif;
uses crt;
var
    bil : integer;

begin
    clrscr;
    write('Masukkan bilangan bulat: '); readln(bil);

    if (bil >= 0) then
        writeln(bil,' adalah bilangan POSITIF')
    else
        writeln(bil,' adalah bilangan NEGATIF');

    readln;
end.
```

1.c Statement if Tiga Kasus atau Lebih

Bentuk ini akan menguji lebih dari satu kondisi, dan akan menampilkan perintah yang berbeda-beda tergantung kondisi mana yang terpenuhi.

```
if (kondisi_1) then
begin
    perintah_jika_kondisi_1_benar;
    ...
end
else if (kondisi_2) then
begin
    perintah_jika_kondisi_2_benar;
    ...
end
else
begin
    perintah;
end;
```

Latihan 3.5 Statement if Tiga Kasus

```
program cek_Positif_Negatif_dan_Nol;
uses crt;
var
    bil : integer;
begin
    clrscr;
    write('Masukkan sebuah bilangan bulat: '); readln(bil);
    if (bil < 0) then
        begin
            writeln(bil, ' merupakan bilangan NEGATIF');
        end
    else if (bil = 0) then
        begin
            writeln('Anda memasukkan bilangan NOL');
        end
    else
        begin
            writeln(bil, ' merupakan bilangan POSITIF');
        end;
    readln;
end.
```

Latihan 3.6 Statement if Lima Kasus

```
program cek_Nilai_Huruf;
uses crt;
var
    NA,NUTS,NUAS : real;
    NI           : char;
begin
    clrscr;
    write('Nilai UTS: '); readln(NUTS);
    write('Nilai UAS: '); readln(NUAS);
    NA := (0.4 * NUTS) + (0.6 * NUAS);
    if (NA >= 80) then
        begin
            NI := 'A';
        end
    else if (NA >= 70) then
        begin
            NI := 'B';
        end
    else if (NA >= 50) then
        begin
            NI := 'C';
        end
    else if (NA >= 30) then
        begin
            NI := 'D';
        end
    else
        begin
            NI := 'E';
        end;
    writeln;
    writeln('Nilai Akhir    : ',NA:0:2);
    writeln('nilai Indeks   : ',NI);
    readln;
end.
```

2. Statement Case

Apabila kondisi yang didefinisikan di dalam *statement if* semakin banyak, maka hal tersebut tentu akan menjadi hal yang cukup kompleks. Oleh karena itu, bahasa Pascal menyediakan alternatif lain untuk melakukan pemilihan yang didasarkan pada nilai-nilai konstanta tertentu, yaitu dengan menggunakan *statement case*.

Bentuk umum *statement case*:

```
case (ekspresi) of
    nilai_konstan1 : statement_1;
    nilai_konstan2 : statement_2;
    ...
    nilai_konstan2 : statement_2
else
    statemen_default;
end;
```

Latihan 3.7 Statement case Cek Indeks Nilai

```
program cek_Nilai_Huruf;
uses crt;
var
    NA,NUTS,NUAS : integer;
    NA_1           : real;
    NI             : char;

begin
    clrscr;

    write('Nilai UTS: '); readln(NUTS);
    write('Nilai UAS: '); readln(NUAS);
    NA_1 := (0.4 * NUTS) + (0.6 * NUAS);
    NA   := round(NA_1);
    case (NA) of
        81..100 : NI := 'A';
        71..80  : NI := 'B';
        51..70  : NI := 'C';
        31..50  : NI := 'D';
        0..30   : NI := 'E';
    else
        begin
            writeln('Range nilai tidak sesuai');
        end;
    end;
    writeln;
    writeln('Nilai Akhir    : ',NA);
    writeln('Nilai Indeks   : ',NI);

    readln;
end.
```

Latihan 3.8 Statement case Cek Hari

```

program cek_Hari_Dengan_Angka;
uses crt;
var
    nomorhari : integer;

begin
    clrscr;

    write('Masukkan nomor hari (1..7): '); readln(nomorhari);
    case (nomorhari) of
        1: writeln('Hari Minggu');
        2: writeln('Hari Senin');
        3: writeln('Hari Selasa');
        4: writeln('Hari Rabu');
        5: writeln('Hari Kamis');
        6: writeln('Hari Jumat');
        7: writeln('Hari Sabtu');
    else
        begin
            writeln('Tidak ada pilihan yang dimasukkan');
        end;
    end;

    readln;
end.

```

IV. Tugas

1. Dengan menggunakan operasi kondisi, buatlah sebuah program untuk mengurutkan 3 buah angka yang di-*input*-kan secara menurun (*descending*).

Input :
Masukkan Nilai 1 : 7
Masukkan Nilai 2 : 2
Masukkan Nilai 3 : 5
Output :
Urutan : 7 5 2

2. Dengan menggunakan *statement case*, buatlah sebuah program untuk memilih menu makanan yang tersedia. Dengan ketentuan sebagai berikut:
 - Minimal terdapat 5 menu makanan yang dapat dipilih pengguna
 - *Input* program berupa kode menu
 - *Output* program merupakan menu yang dipilih, serta biaya yang harus dibayar
 - Menu serta kode yang digunakan tiap praktikan tidak boleh sama
3. Dengan menggunakan operasi kondisi, buatlah sebuah program yang akan menampilkan tahun yang dimasukkan *user* merupakan tahun kabisat atau bukan.

MODUL IV

PERULANGAN DAN PELONCATAN

I. Tujuan

1. Praktikan dapat memahami penggunaan operasi perulangan.
2. Praktikan dapat membuat program yang menggunakan peloncatan dengan benar dalam program.

II. Tugas Persiapan

1. Jelaskan pengertian dari perulangan dan peloncatan!
2. Operator apa yang biasanya digunakan pada operasi perulangan dan jelaskan kenapa!
3. Buatlah *flowchart* untuk menuliskan nama kamu sebanyak x kali!
4. Apa yang kamu ketahui tentang perbedaan perulangan *for*, *while*, dan *repeat*!
5. Apa manfaat dari perulangan dan beri contoh perulangan dalam kehidupan sehari-hari!
6. Sebutkan syarat-syarat suatu perulangan!

III. Teori

Blok pengulangan merupakan suatu blok program yang memiliki mekanisme untuk melakukan *statement* secara berulang. Hal ini tentu akan membuat program yang ditulis menjadi lebih efisien.

1. Statement For

Konstruksi pengulangan *for* pada umumnya digunakan untuk melakukan pengulangan yang banyaknya sudah diketahui secara pasti.

Bentuk umum dari *statement for*:

```
for variable_index := batas_awal to batas_akhir do
    statement_yangakan_diulang;
```

NB: Jika *statement* yang ingin diulang hanya 1 baris, tidak perlu menggunakan begin-end; Tetapi jika *statement* yang ingin diulang lebih dari 1 baris, *statement* harus diletakkan di dalam begin-end.

```
for variable_index := batas_awal to batas_akhir do
begin
    statement_looping_1;
    statement_looping_2;
    ...
    Statement_looping_x;
end;
```

Latihan 4.1 *for Loop* Cetak Kalimat Sebanyak 5 Kali

```
program ulangi_String_5_kali_for;
uses crt;
var
    i : integer;

begin
    clrscr;
    for i := 0 to 4 do
        writeln('ILMU KOMPUTER USU ');
    readln;
end.
```

Latihan 4.2 *for Loop* Hitung Jumlah $1 + 2 + \dots + n$

```
program hitung_Jumlah_1_sampai_n;
uses crt;
var
    i,n,jumlah : integer;

begin
    clrscr;

    write('Masukkan sebuah bilangan bulat: '); readln(n);
    jumlah := 0;
    for i := 1 to n do
        jumlah := jumlah + i;
    writeln('Jumlah dari 1 sampai ',n,' = ', jumlah);

    readln;
end.
```

Latihan 4.3 *for Loop* Cetak Angka 1 sampai 5 Secara Menurun

```
program cetak_1_sampai_5_descending;
uses crt;
var
    i : integer;

begin
    clrscr;
    for i := 5 downto 1 do
        begin
            writeln(i);
        end;
    readln;
end.
```

Latihan 4.4 Nested *for Loop* Cetak Angka Berurut

```
program cetak_Angka_Berurut;
uses crt;
var
    i,j : integer;

begin
    clrscr;

    for i := 1 to 5 do
    begin
        for j := 1 to 5 do
            write(i);
        writeln;
    end;
    writeln;
    for i := 1 to 5 do
    begin
        for j := 1 to 5 do
            write(j);
        writeln;
    end;

    readln;
end.
```

2. Statement While

Berbeda dengan bentuk pengulangan *for*, pada konstruksi pengulangan **while** ini terdapat suatu kondisi yang harus diperiksa terlebih dahulu. Apabila kondisi bernilai *true*, maka *statement* di dalam blok **while** akan dikerjakan, sedangkan jika *false* maka program tidak akan melakukan perulangan.

Bentuk umum *statement while*:

```
while (kondisi) do
    statement;
```

NB: Jika *statement* yang ingin diulang hanya 1 baris, tidak perlu menggunakan *begin-end*; Tetapi jika *statement* yang ingin diulang lebih dari 1 baris, *statement* harus diletakkan di dalam *begin-end*.

```
while (kondisi) do
begin
    statement_1;
    statement_2;
    ...
    statement_x;
end;
```

Latihan 4.5 *while* Cetak Kalimat Sebanyak 5 Kali

```
program ulangi_String_5_kali_while;
uses crt;
var
    i : integer;
begin
    clrscr;

    i := 1;
    while (i <= 5) do
        begin
            writeln('Ilmu Komputer USU');
            i := i + 1;
        end;

    readln;
end.
```

Latihan 4.6 *while* Cetak Angka 1 sampai 20 yang Ganjil dengan Fungsi *inc* (*increment*)

```
program cetak_1_sampai_20_Ganjil_while_inc;
uses crt;
var
    i : integer;
begin
    clrscr;

    i := 1;
    while (i <= 20) do
        begin
            write(i, ' ');
            inc(i,2); //sama dengan i:=i+2;
        end;

    readln;
end.
```

3. Statement Repeat-Until

Pernyataan *repeat-until* digunakan untuk mengulang (*repeat*) suatu pernyataan atau blok pernyataan terus-menerus sampai (*until*) kondisi ungkapannya tidak terpenuhi (*false*).

Struktur penggunaan perulangan *repeat-until*.

```
repeat
    pernyataan yang diulang;
until (ekspresi logika);
```

Latihan 4.7 *repeat-until* Cetak Angka 1 sampai 5

```
program cetak_1_sampai_5_repeat_until;
uses crt;
var
    i : integer;

begin
    clrscr;

    i := 0;
    repeat
        i := i + 1;
        writeln(i);
    until (i = 5);

    readln;
end.
```

4. Peloncatan (Penggunaan *Label*)

Statement peloncatan merupakan *statement* yang digunakan untuk memaksa program agar meloncat atau menuju *statement* tertentu yang kita inginkan. Dalam bahasa Pascal terdapat sebuah statement yang sering digunakan untuk melakukan hal tersebut yaitu *statement goto* dan empat buah prosedur yaitu *break*, *continue*, *exit* dan *halt*.

4.a label dan *Statement goto*

Untuk menggunakan *statement goto*, sebelumnya kita harus mendefinisikan sebuah label dengan menggunakan kata kunci label. Di sini label akan berperan sebagai baris yang akan dituju. Dengan kata lain, *statement goto* akan memindahkan program secara langsung ke lokasi yang ditandai oleh label yang telah didefinisikan.

Latihan 4.8 Statement *goto*

```
program statement_goto;
uses crt;
label destination;

begin
    clrscr;

    writeln('Statement pertama');
    goto destination;
    writeln('Statement kedua');
    writeln('Statement ketiga');
    destination: writeln('Ini adalah statement yang akan dituju');

    readln;
end. //what happen??
```

4.b Prosedur *break*

Prosedur ***break*** digunakan untuk meloncat dari proses pengulangan. Artinya, prosedur ***break*** akan memaksa program untuk menghentikan pengulangan dan akan langsung menuju ke *statement* selanjutnya yang terdapat di bawah blok pengulangan

Latihan 4.9 Prosedur *break*

```
program prosedur_break;
uses crt;
var
  i : integer;

begin
  clrscr;

  writeln('Ini adalah statement sebelum looping');
  writeln;

  for i := 1 to 100 do
    begin
      if (i > 3) then
        break;
      writeln('Statement ke-',i);
    end;

  writeln;
  writeln('Apa yang terjadi??');

  readln;
end.
```

4.c Prosedur *continue*

Prosedur ***continue*** merupakan kebalikan dari prosedur ***break***, yaitu berguna untuk melanjutkan pengulangan dengan cara meloncat ke bagian awal pengulangan. Pada umumnya prosedur ***continue*** digunakan di dalam kasus dimana kondisi pengulangannya selalu bernilai *true*.

Latihan 4.10 Prosedur *continue*

```
program prosedur_continue;
uses crt;
var
  x : integer;

begin
  clrscr;

  while (true) do
    begin
      write('Masukkan bilangan bulat negatif: '); readln(x);
      if (x > -1) then
        continue
      else
        writeln('Anda telah memasukkan bilangan: ',x);
    end;

  readln;
end. //apa yang terjadi? :D
```

4.d Prosedur exit

Prosedur **exit** digunakan untuk keluar dari suatu blok program. Namun apabila prosedur **exit** kita gunakan di dalam program utama, maka prosedur tersebut akan menyebabkan program berhenti.

Latihan 4.11 Prosedur exit

```
program prosedur_exit;
uses crt;

procedure contoh;
begin
    writeln('Statement 1 di dalam prosedur');
    exit;
    writeln('Statement 2 di dalam prosedur');
end;

begin
    clrscr;

    writeln('Statement 1 di dalam program utama');
    contoh;
    writeln('Statement 2 di dalam program utama');

    readln;
end.
```

4.e Prosedur halt

Berbeda dengan prosedur **exit** yang digunakan untuk keluar dari blok program, prosedur **halt** berguna untuk keluar dari program. Prosedur **halt** memiliki parameter berupa nilai **0** dan **1** yang bersifat **opsional**. *Statement* yang terdapat di bawah prosedur **halt** tidak akan dieksekusi oleh program karena program akan langsung dihentikan.

Latihan 4.12 Prosedur halt

```
program prosedur_exit;
uses crt;

begin
    clrscr;

    writeln('Statement sebelum pemanggilan prosedur halt');
    halt(1);
    writeln('Statement sesudah pemanggilan prosedur halt');

    readln;
end.
```

IV. Tugas

1. Buatlah sebuah program untuk menghitung jumlah nilai dari 1 sampai n!

Input	Output
n = 4	1 + 2 + 3 + 4 = 10 1 + 2 + 3 = 6 1 + 2 = 3 1 = 1

2. Buatlah sebuah program untuk membuat tampilan seperti berikut!

```

*
2 2
* * *
4 4 4 4
* * * * *

```

3. Buatlah sebuah program untuk membuat tampilan seperti berikut!

```

*
* * *
* * * * *
* * * * * *
* * * *
*

```

4. Buatlah sebuah program yang menentukan apakah angka tersebut prima dengan *range* 1 - 50!
5. Buatlah sebuah program perkalian angka 1 dari 1 - 100. apabila *output* dapat dibagi 3 dan 5 *output* tersebut tidak akan ditampilkan.

MODUL V ARRAY

I. Tujuan

1. Praktikan memahami *array* dan penggunaannya.
2. Praktikan dapat membuat program yang menggunakan *array*.

II. Tugas Persiapan

1. Apa yang Anda ketahui mengenai *array/larik*?
2. Apa fungsi *array* dalam dunia pemrograman?
3. Apa fungsi indeks pada *array*?
4. Apa keterkaitan antara perulangan dengan *array*?
5. Sebutkan beberapa contoh untuk implementasi dari *array* dalam kehidupan?

III. Teori Array

Array (larik) adalah tipe data terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe sama. Komponen-komponen tersebut disebut sebagai komponen *type*, larik mempunyai jumlah komponen yang jumlahnya tetap. Banyaknya komponen dalam larik ditunjukkan oleh suatu *index* yang dimulai dari 1, dimana tiap komponen di *array* dapat diakses dengan menunjukkan nilai *index-nya* atau *subscript*.

Array dapat bertipe data sederhana seperti *byte*, *word*, *integer*, *real*, *boolean*, *char*, *string* dan tipe data *scalar* atau *subrange*. Tipe larik mengartikan isi dari larik atau komponen-komponennya mempunyai nilai dengan tipe data tersebut.

Agar lebih jelas, perhatikan contoh berikut.

Index	1	2	3	4
X	31	21	23	12

$$\begin{array}{ll} X[1] = 31 & X[3] = 23 \\ X[2] = 21 & X[4] = 12 \end{array}$$

Tipe data larik dapat berupa larik satu dimensi, dua dimensi, tiga dimensi atau banyak dimensi.

Bentuk Umum Larik Satu Dimensi:

```
var
  X : array[1..50] of integer;
```

Contoh Program:

```
program Contoh_Array_Input;
uses crt;
var Bilangan : array[1..20] of integer;
begin
  clrscr;
  Bilangan[1] := 3;
  Bilangan[2] := 29;
  Bilangan[3] := 30;
  Bilangan[4] := 31;
  Bilangan[5] := 23;
  writeln('Nilai variabel bilangan ke 3 = ',Bilangan[3]);
  writeln('Nilai variabel bilangan ke 5 = ',Bilangan[5]);
  readln;
end.
```

Output

Nilai variabel bilangan ke 3 = 30
 Nilai variabel bilangan ke 5 = 23

Bentuk Umum Larik Multi Dimensi:

Larik multi dimensi di bawah memiliki 3 dimensi.

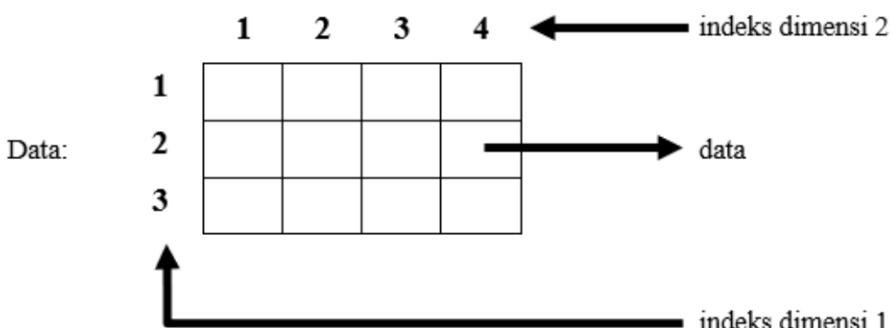
```
var
  X : array[1..50,1..20,1..5] of integer;
```

Pengaksesan larik di atas: **X[indeks1, indeks2, indeks3];**

Contoh :

```
data : array [1..3,1..4] of integer;
```

Artinya terdapat variabel yang bernama data yang bertipe integer dan dideklarasikan dengan larik dua dimensi, dengan jumlah maksimal baris = 3, dan kolom = 4. Sehingga dapat diilustrasikan sebagai berikut:



Berikut merupakan contoh sederhana dari pengaplikasian array dalam **matriks**.

1. Struktur Matriks

```
program strukturMatriks(input,output);
uses crt;
type
    larik = array[1..10,1..10] of integer;
var
    matriks1,matriks2 : larik;
begin
    clrscr;
    matriks1[1,1] := 2;
    matriks1[1,2] := 4;
    matriks2[2,1] := 9;
    matriks2[2,2] := 5;
    clrscr;
    writeln(matriks1[1,1],', ',matriks1[1,2]);
    writeln(matriks2[2,1],', ',matriks2[2,2]);
    readln;
end.
```

2. Penjumlahan Matriks

```
program penjumlahanMatriks(input,output);
uses crt;
type
    larik = array[1..10,1..10] of integer;
var
    matriks1,matriks2,hasil : larik;
    jum,i,j,k : integer;
begin
    clrscr;
    matriks1[1,1] := 2;
    matriks1[1,2] := 4;
    matriks1[2,1] := 8;
    matriks1[2,2] := 6;
    matriks2[1,1] := 7;
    matriks2[1,2] := 1;
    matriks2[2,1] := 9;
    matriks2[2,2] := 0;
    clrscr;
    writeln('Program Penjumlahan Matriks');
    writeln;
    writeln('Matriks Pertama');
    for i:=1 to 2 do
    begin
        for j:=1 to 2 do
        begin
            write(matriks1[i,j],', ');
        end;
        writeln;
    end;
    writeln('Matriks Kedua');
    for i:=1 to 2 do
    begin
```

```

        for j:=1 to 2 do
        begin
            write(matriks2[i,j], ' ');
        end;
        writeln;
    end;

    {proses penjumlahan}
    for i:=1 to 2 do
    begin
        for j:=1 to 2 do
        begin
            hasil[i,j] := matriks1[i,j] + matriks2[i,j];
        end;
    end;

    writeln('Hasil Penjumlahan');
    for i:=1 to 2 do
    begin
        for j:=1 to 2 do
        begin
            write(hasil[i,j], ' ');
        end;
        writeln;
    end;
    readln;
end.

```

3. Perkalian Matriks

```

program perkalianMatriks(input,output);
uses crt;
type
    larik = array[1..10,1..10] of integer;
var
    matriks1,matriks2,hasil : larik;
    i,j,k : integer;
    jumlah : integer;
begin
    clrscr;
    matriks1[1,1] := 2;
    matriks1[1,2] := 4;
    matriks1[2,1] := 8;
    matriks1[2,2] := 6;
    matriks2[1,1] := 7;
    matriks2[1,2] := 1;
    matriks2[2,1] := 9;
    matriks2[2,2] := 0;
    clrscr;
    writeln('Program Perkalian Matriks');
    writeln;
    writeln('Matriks Pertama');

```

```
for i:=1 to 2 do
begin
    for j:=1 to 2 do
    begin
        write(matriks1[i,j], ' ');
    end;
    writeln;
end;

writeln('Matriks Kedua');
for i:=1 to 2 do
begin
    for j:=1 to 2 do
    begin
        write(matriks2[i,j], ' ');
    end;
    writeln;
end;

{proses perkalian}

jumlah := 0;
for i:=1 to 2 do
begin
    for j:=1 to 2 do
    begin
        for k := 1 to 2 do
        begin
            jumlah := jumlah + matriks1[i][k] * matriks2[k][j];
        end;
        hasil [i, j] := jumlah;
        jumlah := 0;
        end;
    end;
end;

writeln('Hasil Perkalian');
for i:=1 to 2 do
begin
    for j:=1 to 2 do
    begin
        write(hasil[i,j], ' ');
    end;
    writeln;
end;
readln;
end.
```

IV. Tugas dan Latihan

- Buatlah program yang menyimpan data urutan antrian n ($1 \leq n \leq 10$) mahasiswa (ditandai dengan NIM) yang sedang menunggu pesanan di kantin mulai dari yang paling depan ke belakang. Kemudian tampilkan urutan antiran n mahasiswa tersebut (ditandai dengan NIM) dari belakang ke depan.

Input (dari depan ke belakang)	Output (dari belakang ke depan)
5 033 001 054 090 111	111 090 054 001 033

- Buatlah program penjumlahan matriks, dengan jumlah kolom dan baris matriks ke-1 dan ke-2 ditentukan oleh *user*. Program juga dapat menentukan apakah kedua matriks tersebut dapat dijumlahkan atau tidak.

Note : jumlah kolom dan baris matriks maksimal sebanyak 5.

- Buatlah program yang menampung nilai UAS dari n ($1 \leq n \leq 10$) mahasiswa. Kemudian tampilkan nilai n mahasiswa tersebut yang diurutkan dari terendah ke tertinggi.
- Buatlah sebuah program yang dapat menampung nilai-nilai n ($1 \leq n \leq 5$) mahasiswa (terdiri dari nilai tugas, nilai kuis, nilai UTS, dan nilai UAS). Kemudian tentukan apakah mahasiswa lulus atau tidak (lulus jika nilai rata-rata ≥ 70).

Input:

- Masukkan jumlah mahasiswa.
- Masukkan nilai-nilai mahasiswa ke-i.

Output: Tampilkan nilai-nilai dari n mahasiswa beserta nilai rata-rata dan keterangan lulus/tidak.

MODUL VI

PROSEDUR DAN FUNGSI

I. Tujuan

1. Praktikan memahami pembuatan dan pemakaian suatu Prosedur dan Fungsi.
2. Praktikan dapat membuat program dengan menggunakan Prosedur dan Fungsi.
3. Praktikan dapat membedakan Prosedur dengan Fungsi.

II. Tugas Persiapan

1. Jelaskan *functional programming*, *procedural programming*, dan *object oriented programming*!
2. Jelaskan perbedaan fungsi dan prosedur!
3. Jelaskan parameter dan kegunaannya!
4. Tuliskan minimal 5 fungsi dalam bidang matematika, fisika, dan kimia. (tiap praktikan tidak boleh sama).

III. Teori Procedure

Prosedur adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai subprogram (program bagian). Prosedur diawali dengan keyword procedure di dalam bagian deklarasi prosedur.

1. Struktur Procedure

1.a Struktur penggunaan prosedur tanpa parameter

```
program {judul program};

procedure {judul prosedur};
begin
    {blok pernyataan prosedur}
end;

begin
    {blok pernyataan program utama}
end.
```

Latihan 6.1

```
procedure Stars;
begin
    writeln('*****');
end;

begin
    stars;
    writeln('Hello Pascal!');
    stars;
end.
```

Latihan 6.2

```

program ProsedurDenganParameter;
uses crt;
type
    Huruf = String[15];

procedure Stars(Nama : Huruf);
var
    k,panjang : integer;
begin
    k := 0;
    panjang := length(Nama) + 6;
    repeat
        write('*');
        k := k + 1;
    until k > panjang;
    writeln;
end;

var
    Nama : Huruf;

begin
    write('Nama Anda = '); readln(Nama);
    Stars(Nama);
    writeln('Hello ', Nama, '!'); Stars(Nama);
    readln();
end.

```

IV. Teori Fungsi

Definisi fungsi sebenarnya sama dengan prosedur. Perbedaannya pada fungsi terdapat pengembalian nilai, sehingga pada saat pemanggilan, fungsi dapat langsung digunakan untuk mengisi sebuah ekspresi. Berbeda dengan prosedur yang didefinisikan dengan kata kunci *procedure*, fungsi didefinisikan dengan kata kunci *function*.

Bentuk umum fungsi:

```

function nama_fungsi(parameter1:tipedata1;parameter2:tipedata2,...):tipedata;
const
    {daftar konstanta lokal}
var
    {daftar pendeklarasian variable lokal}

Begin
    {kode program yang akan ditulis}
    ...
    NamaFungsi := nilai_kembalian; {bagian ini harus diINGAT!}
end;

```

Beberapa contoh pendefinisian fungsi sederhana yang akan mengalikan dua buah bilangan bulat:

```

function Kali(x,y:integer):longint;
begin
    Kali := x*y;
end;

```

Bisa juga seperti ini:

```
function kali(x,y: integer):longint;
var
    hasil : longint;
begin
    hasil := x*y;
    kali := hasil;
end;
```

1. Parameter

Parameter merupakan suatu nilai atau referensi yang dilewatkan ke dalam rutin tertentu dan kehadirannya akan mempengaruhi proses maupun nilai yang terdapat di dalam rutin itu sendiri. Parameter ditempatkan di dalam tanda kurung setelah nama rutin bersangkutan. Setiap parameter yang dilewatkan harus memiliki tipe data tersendiri.

Secara teori, parameter dibedakan menjadi dua yaitu:

- Parameter formal → parameter yang terdapat pada saat pembuatan prosedur/fungsi.
- Parameter aktual → parameter yang terdapat pada saat pemanggilan prosedur/fungsi.

Menurut fungsinya, parameter dibedakan menjadi tiga jenis yaitu sebagai masukan, keluaran, masukan/keluaran.

2. Parameter Masukan

Secara *default* parameter yang digunakan di dalam sebuah rutin akan berperan sebagai masukan, artinya nilai yang disimpan di dalam parameter tersebut akan dijadikan sebagai *input* (masukan) untuk melakukan proses yang terdapat di dalam rutin bersangkutan.

Perhatikan contoh:

Latihan 6.3

```
program parametermasukan;
uses crt;

function jumlahkan(x,y: integer): integer;
begin
    Jumlahkan := x+y; {x dan y bertindak sebagai parameter masukan}
end;

var
    a,b,hasil : integer;

begin
    clrscr;
    a := 100;
    b := 200;
    hasil := jumlahkan(a,b);
    writeln('Hasil= ',hasil);
    readln;
end.
```

3. Parameter Keluaran

Parameter ini berfungsi sebagai penampung nilai hasil proses yang dilakukan oleh suatu subroutine (*subroutine*). Parameter jenis ini biasanya diimplementasikan di dalam sebuah prosedur sebagai nilai kembalian. Hal ini disebabkan karena prosedur tidak dapat mengembalikan nilai secara langsung seperti yang dilakukan oleh fungsi.

Perhatikan contoh:

Latihan 6.4

```
program parameterkeluaran;
uses crt;

procedure Kali(x,y:integer; var hasil: integer);
begin
    hasil := x*y;
end;
{x dan y parameter masukan, sementara hasil adalah parameter keluaran}

var
    a,b,c: integer;

begin
    clrscr;
    a := 3;
    b := 5;
    kali(a,b,c);
    writeln('Hasil= ',c);
    readln;
end.
```

4. Parameter Masukan/Keluaran

Parameter jenis ini merupakan gabungan dari jenis pertama dan kedua, yaitu berperan sebagai masukan sekaligus sebagai keluaran. Untuk mendefinisikan parameter sebagai parameter masukan/keluaran maka kita harus melewatkannya berdasarkan referensi.

Perhatikan contoh:

Latihan 6.5

```
program parametermaskel;
uses crt;

procedure tambahsatu(var x:integer);
begin
    x := x+1;
end;

var
    a : integer;

begin
    a := 10;
    tambahsatu(a);
    writeln(a);
    readln();
end.
```

V. Tugas

1. Buat program menghitung luas segitiga dengan menggunakan fungsi dengan parameter masukan!

Input	Output
Alas = 7 Tinggi = 4	Luas Segitiga = 14

2. Buat program menghitung rata-rata dari sejumlah bilangan yang di-input-kan operator dengan fungsi!

Input	Output
Banyak Data : 5 1 2 3 4 5	Rata-rata = 3.00

3. Buat program dari $n!$ secara rekursi (*recursive*)!

Input	Output
Nilai : 7	Hasil = 5040

4. Buat program untuk menentukan apakah suatu bilangan merupakan bilangan prima!

Input	Output
6	Non-Prima
5	Prima

5. Buatlah program menggunakan fungsi untuk menampilkan deret bilangan berikut

- :a. 1,1,2,3,5,8,13,21,34,....
- b. 1,2,4,8,16,32,....
- c. 1,4,9,16,25,....
- d. 2,6,12,20,30,....

Input	Output
3	1, 1, 2 1, 2, 4 1, 4, 9 2, 6, 12
6	1, 1, 2, 3, 5, 8 1, 2, 4, 8, 16, 32 1, 4, 9, 16, 25, 36 2, 6, 12, 20, 30, 42

MODUL VII

RECORD

I. Tujuan

1. Praktikan memahami konsep dasar *record* dalam Pascal
2. Praktikan dapat menggunakan *record* dalam pemrograman

II. Tugas Persiapan

1. Tuliskan dengan bahasamu sendiri, apa yang kamu ketahui tentang *record*!
2. Tuliskan beberapa fungsi dan kegunaan *record*!
3. Tuliskan fungsi pernyataan WITH pada *record*, dan jelaskan kelebihan dan kekurangannya!
4. Apa yang Anda ketahui tentang *array record* dan *nested record* serta tuliskan kegunaannya masing-masing menurut pendapat kalian!

III. Teori

Record adalah jenis tipe data terstruktur yang berisi beberapa data, yang masing-masing dapat berlainan tipe. Suatu tipe Record dideklarasikan dengan bentuk sebagai berikut:

```
type identifier = Record
    nama_field1 : tipe1;
    nama_field2 : tipe2;
    ...
    nama_fieldn : tipen;
end;
```

Masing-masing data *field* dapat berupa satu atau beberapa nama pengenal dan masing-masing dinamakan *field*.

Contoh Penggunaan Record dalam
Pascal:Latihan 7.1

```
program Record1;
uses crt;

type biodata = Record
    nama : string[25];
    NIM : string[10];
    umur : integer;
end;

var
    data : biodata;

begin
    clrscr;
    write ('Masukkan Nama Anda : '); readln(data.nama);
    write ('Masukkan NIM Anda : '); readln(data.NIM);
    write ('Masukkan Umur Anda : '); readln(data.umur);
    clrscr;
    writeln('Nama Anda : ',data.nama);
    writeln('NIM Anda : ',data.NIM);
    writeln('Umur Anda : ',data.umur);
    readln;
end.
```

Contoh di atas merupakan contoh *record* sederhana untuk meng-*input* data mahasiswa. Dalam *record*, penampilan data tidak boleh dilakukan secara langsung, contohnya **writeln(nama)**, yang harus kita lakukan adalah kita harus menuliskan variabel dan *field* yang akan ditampilkan, contoh **writeln(mahasiswa.nama)**.

IV. Penugasan Antar Record

Penugasan antar *record* merupakan penyalinan yang dilakukan dari *record* satu ke *record* yang lain, tetapi *record* yang disalin dan yang menyalin bertipe yang sama.

Contoh:

Latihan 7.2

```

type recBio = Record
    Nama      : string[25];
    NIM       : integer;
    NoHp     : string[12];
end;

var
    Bio1,Bio2 : recBio;

begin
    clrscr;
    Bio1.Nama := 'Nama Saya';
    Bio1.NIM  := 'NIM Saya';
    Bio1.NoHP := 'No. HP Saya';

    Bio2 := Bio1; (*Proses Menyalin Dari Record Bio1 ke Bio2*)

    {Menampilkan Data}
    writeln('Nama   : ',Bio2>Nama);
    writeln('NIM    : ',Bio2>NIM);
    writeln('No. HP : ',Bio2>NoHP);
end.

```

Contoh di atas akan menampilkan isi dari *record* dari Bio1.

V. Pernyataan With

Pernyataan *with* di dalam *record* berfungsi untuk menyederhanakan atau menyingkat penulisan pernyataan. Bentuk penulisan:

```

WITH nama_record Do
begin
    {pernyataan}
end;

```

Contoh:
Latihan 7.3

```

type recBio = Record
    Nama      : string[25];
    NIM       : integer;
    NoHp     : string[12];
End;

var
    Biol,Bio2   : recBio;

begin
    clrscr;
    with Biol do
    begin
        Nama := 'Nama saya';
        Alamat := 'NIM saya';
        NoHP := 'No. HP saya';
    end;

    Bio2 := Biol;

    {Menampilkan Data}
    with bio2 do
    begin
        writeln('Nama    : ',Nama);
        writeln('NIM     : ',NIM);
        writeln('NO. HP : ',NoHP);
    end;

    readln;
end.
```

VI. Array Record

Elemen dari suatu *array* juga bisa berupa *record*, yang menampung beberapa data sekaligus (data pertama hingga data ke n).

Penulisan *array record*:

```

type Identifier = Record
    Nama_field1 : tipe1;
    Nama_field2 : tipe2;
    ...
    Nama_fieldn : tipen;
End;

var
    variabel : array[1..n] of Identifier; //array record
```

Contoh *array record*:

Latihan 7.4

```
program Recordn;
uses crt;

type KRS = Record
    matkul : string[25];
    SKS     : integer;
    Kode    : string[10];
end;

var
    data   : array[1..10] of KRS;
    n,i    : integer;

begin
    clrscr;
    write('Masukkan Banyak Mata Kuliah Yang Diambil : '); readln(n);
    clrscr;

    for i:=1 to n do
    begin
        writeln('Masukkan Data ke-',i);
        with data[i] do
        begin
            write('Masukkan Nama Matkul : '); readln(matkul);
            write('Masukkan Jumlah SKS : '); readln(SKS);
            write('Masukkan Kode SKS      : '); readln(Kode);
            clrscr;
        end;
    end;
    clrscr;

    for i:=1 to n do
    begin
        writeln('Data ke-',i);
        with data[i] do
        begin
            writeln('Nama Mata Kuliah : ',matkul);
            writeln('Jumlah SKS       : ',SKS);
            writeln('Kode Mata Kuliah : ',kode);
        end;
        writeln;
    end;

    readln;
end.
```

VII. Record dalam Record (Nested Record)

Record di dalam *record* adalah *record* yang mempunyai tipe *record* lain. Contoh penulisan:

```
Type
record1 = Record
    Nama_field1 : tipe1;
    Nama_field2 : tipe2;
    ...
    Nama_fieldn : tipen;
end;

record2 = Record
    Namal : Reord1;
    Nama2 : identifier1;
    Nama3 : Identifier2;
end;

var
    variabel : record;
```

Contoh Penggunaan Record dalam Record:**Latihan 7.5**

```
program recordinrecord;
uses crt;

type
date = Record
    tanggal : string[30];
    bulan   : integer;
    tahun   : string[25];
end;
album = Record
    penyanyi : char;
    lagu     : string[25];
    rilis    : date;
end;

var
    data      : album;

begin
    clrscr;
    with data do
    begin
        write ('Masukkan Nama Penyanyi : '); readln(penyanyi);
        write ('Masukkan Nama Lagu      : '); readln(lagu);
        writeln('Masukkan Tanggal Rilis   ');
        with rilis do
        begin
            write('- Tanggal : '); readln(tanggal);
            write('- Bulan   : '); readln(bulan);
            write('- Tahun   : '); readln(tahun);
        end;
    end;
    clrscr;
    writeln('Data yang Anda Masukkan');
    with data do
    begin
        writeln('Nama    : ',penyanyi);
        writeln('Lagu   : ',lagu);
        writeln('Tanggal Rilis : ');
        with rilis do
        begin
            writeln('Tanggal : ',tanggal);
            writeln('Bulan   : ',bulan);
            writeln('Tahun   : ',tahun);
        end;
    end;
    readln;
end.
```

VIII. Tugas

1. Buatlah sebuah program untuk meng-*input*-kan beberapa biodata mahasiswa (nama, nim, alamat, No. HP, tempat lahir, tanggal lahir (tanggal, bulan (angka), tahun)). Kemudian, tampilkan data-data yang telah di-*input*-kan tersebut. (NB: gunakan perulangan)!
2. Buatlah sebuah program untuk menampilkan data beberapa mahasiswa dengan sebuah kondisi(if).
3. Buatlah sebuah program untuk meng-*input*-kan hasil studi beberapa mahasiswa yang berisi nama, NIM, dan jumlah mata kuliah yang diambil. Kemudian program akan meminta nama mata kuliah, jumlah SKS, dan nilai, sesuai dengan jumlah mata kuliah yang di-*input*-kan pada setiap mahasiswa. Program akan menampilkan kembali data-data hasil study mahasiswa tersebut satu persatu (menunggu *user*, menekan tombol, kemudian data selanjutnya di tampilkan) hingga semua data selesai ditampilkan.

A = 4, B+ = 3.5, B = 3, C+ = 2.5, C = 2, D = 1, E = 0.

Contoh hasil program:

Nama : Nama Anda NIM : NIM Anda					
<hr/>					
No	Mata Kuliah	SKS	Nilai	Skor	
1	Algoritma	3	B+	10.5	
2	Kalkulus	3	B	9	
3	Logika Komputer	3	A	12	
<hr/>					
TOTAL				31.5	
<hr/>					
Jumlah SKS : 9					
IP : 3.50					
Tekan sembarang tombol untuk melihat data selanjutnya..					

MODUL VIII

POINTER

I. Tujuan

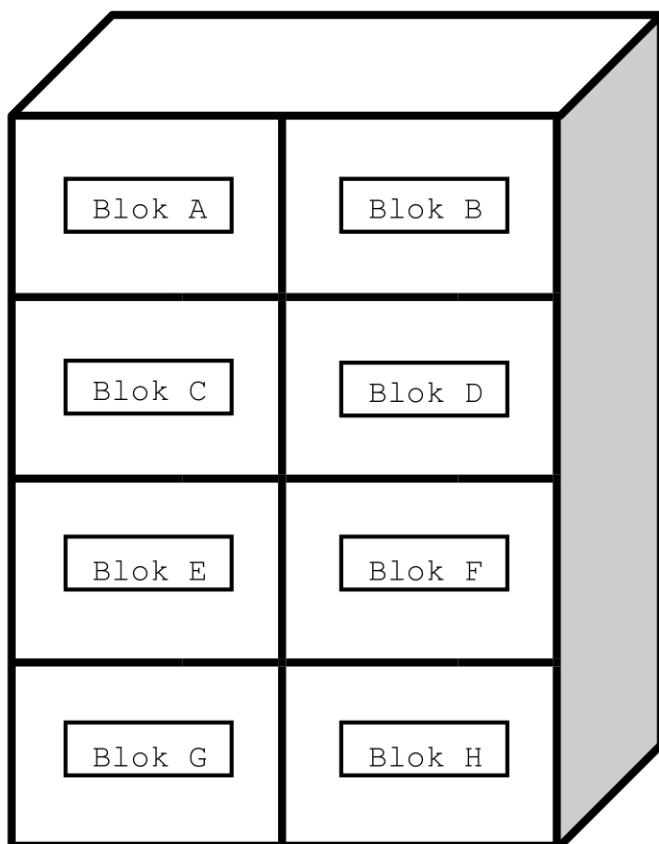
1. Praktikan memahami konsep dasar *pointer* dan penggunaannya.
2. Praktikan bisa membuat program yang menggunakan *pointer*.

II. Tugas Persiapan

1. Apakah perbedaan *variable* biasa dengan *variable pointer*?
2. Jelaskan apa yang dimaksud dengan alamat memori dan blok memori?
3. Apakah *pointer* perlu dipelajari? Jelaskan dengan *argument*!

III. Teori

Pointer merupakan jenis perintah yang disediakan oleh bahasa pemrograman untuk melakukan operasi terhadap suatu alamat memori seperti penunjukan alamat, pengaksesan alamat dan sebagainya. Tipe data *pointer* bersifat dinamis, *variable* dapat diletak ke variabel *pointer* hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan lagi isi dari variabel *pointer* tersebut dapat dihapus. Setiap kita membuat sebuah variabel, Pascal akan membuat sebuah ruang di *memory* komputer. Ruang ini memiliki sebuah alamat yang dapat kita tampilkan dengan variabel *pointer*. Pendeklarasian variabel *pointer* tidak berbeda jauh dengan pendeklarasian *variable* biasa, hanya perlu menambahkan simbol “^” sebelum tipe datanya.



Analogi Pointer

Di Analogikan sebagai lemari yang memiliki stempel alamat dimana stempel alamat tersebut adalah alamat *pointer* dan dalam lemari adalah isi nilai dari tiap *pointer*.

Bentuk umum deklarasi pointer:

```
<nama_var> : ^<tipe_data>
```

Perbedaan:

Variabel Biasa	Variabel Pointer
a = isi data	a^ = isi data
@ = alamat data	a = alamat data

Contoh:

```

var
    nama : ^string[30];

type
    data = record
        no      : integer;
        pilihan : char;
    end;

RecordData : ^data;

```

Dalam penggunaannya, *pointer* menggunakan 2 buah operator yaitu operator `^` untuk mengambil nilai dari alamat yang ditunjuk oleh *pointer* dan operator `@` untuk mengambil alamat dari sebuah *variable*.

Contoh Program:

```

program tipe_pointer;
uses crt;

var
    angka : integer;
    ptr   : ^integer;
    i     : ^word;

begin
    clrscr;

    angka := 15;
    writeln('Nilai dari variable angka: ',angka);

    ptr := @angka;
    writeln('Nilai dari pointer : ',ptr^);

    i := addr(ptr);
    writeln('Alamat memory yang digunakan adalah : ', i^);

    readln;
end.

```

Pointer Tanpa Tipe

Menurut keadaan *default*, *pointer* hanya dapat menunjuk ke alamat yang tipe datanya sama dengan tipe data yang dideklarasikan pada *pointer*. Seperti pada contoh program *pointer1*, *pointer* *ptr* tersebut hanya dapat menunjuk ke alamat-alamat memori yang menyimpan data dengan tipe integer saja, artinya, apabila kita memiliki variabel *c* yang bertipe data char, maka kita tidak dapat memerintahkan *pointer* *ptr* untuk menunjuk ke alamat dari variabel *c*. Untuk mengatasi kasus tersebut, bahasa Pascal telah menyediakan tipe generik untuk *pointer* sehingga *pointer* tersebut dapat menunjuk ke alamat yang berisi data dengan tipe apa saja. *Pointer* yang dideklarasikan dengan tipe generik ini sering disebut dengan *pointer tanpa tipe*. Contoh *pointer* tipe generik:

```

var
  ptr : pointer;
  X   : Integer;
  Y   : real;
  Z   : char;

begin
  ptr := @x;
  ptr := @y;
  ptr := @z;
end.

```

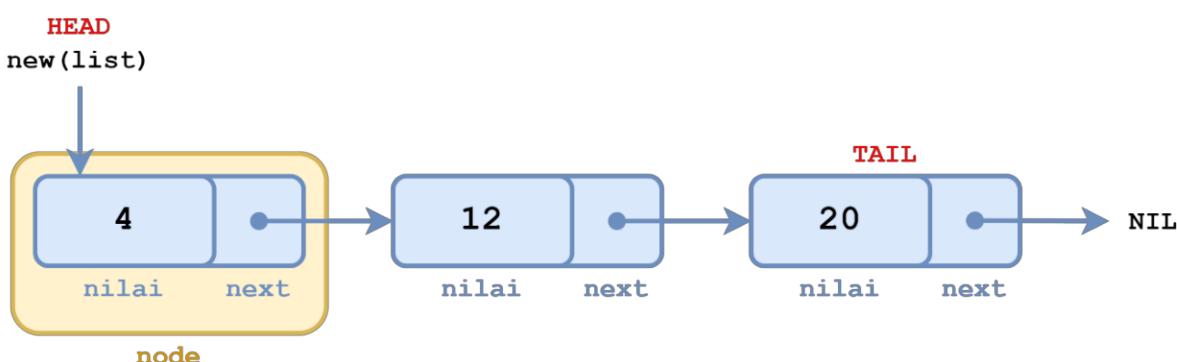
Contoh Implementasi Pointer

Ketika membuat program, kita tidak harus selalu menggunakan *pointer*. Gunakanlah *pointer* ketika memang diperlukan. Mengapa begitu? Sebab variabel *pointer* bisa dikatakan sebagai variabel yang spesial. *Pointer* dapat digunakan sebagai referensi atau “penunjuk” lokasi alamat suatu variabel pada memori (RAM/*Random Access Memory*).

Sekarang pertanyaannya, mengapa *pointer* diciptakan? Padahal *pointer* tidak selalu akan digunakan saat membuat program. Pada zaman dahulu, memori komputer sangatlah terbatas jumlahnya. Jika kita bandingkan dengan komputer modern zaman sekarang yang memiliki memori (RAM) paling tidak 2 GB, dan ada beberapa yang 4 GB, 8 GB, 16 GB, dan seterusnya. Berbeda dengan komputer zaman dahulu yang memiliki memori hanya beberapa kiloByte (kB). Bahkan pada zaman dahulu, hanya dengan bermodal memori (RAM) sebesar 4 kB saja sudah bisa mengirimkan seseorang ke Bulan. Keren bukan?

Dalam pemrograman, terdapat beberapa kasus di mana kita lebih baik menggunakan *pointer* daripada variabel biasa. Contohnya adalah ketika kita ingin membuat struktur data senarai berantai atau dalam Bahasa Inggris, yaitu *linked list*. *Linked list* merupakan suatu struktur data yang membentuk unaian yang saling berhubungan. Nah, setiap unaian tersebut memanfaatkan memori untuk saling menghubungkannya. Sebuah data pada *linked list* biasa disebut dengan ***node*** atau simpul. *Node* tersebut memiliki ***nilai*** dan ***next*** sebagai alamat *pointer* yang digunakan untuk menunjuk kepada *node* baru selanjutnya. *Node* pertama disebut sebagai ***head*** atau kepala *linked list*, dan *node* terakhir disebut sebagai ***tail*** atau ekor dari *linked list*.

Masih bingung? Terdengar sulit dan tampaknya sangat kompleks? Tenang, jangan khawatir, contoh implementasi ini hanya sebagai pengenalan tambahan saja, dan akan dibahas di semester ke depannya. Yuk, coba perhatikan gambar ilustrasi dari *linked list* di bawah ini.



- Sebuah *linked list* baru akan dibentuk terlebih dahulu dengan mendefinisikan bahwa *list* tersebut memiliki komponen (variabel), yaitu **nilai**, dan **next**.
- Sebuah untaian *linked list* disebut dengan *node* atau simpul. Komponen **nilai** diisi dengan sebuah data yaitu nilai 4 dengan tipe **integer**, dan komponen **next** merupakan alamat *pointer* dari *node* tersebut yang digunakan untuk menghubungkan (me-link) ke *node* selanjutnya.
- *Node* selanjutnya akan terhubung/terkaitkan dengan *node* sebelumnya, dan diisi dengan data yang baru, yaitu nilai 12 dengan tipe data **integer**. Begitu juga dengan *node* selanjutnya.
- *Node* pertama dalam sebuah *linked list* disebut dengan *head* (kepala), dan *node* terakhir dalam sebuah *linked list* disebut dengan *tail* (ekor).
- Referensi *pointer* **next** pada *node* terakhir (*tail*) pada sebuah *linked list* akan bernilai **NIL** atau **null**, yaitu kosong atau “tidak ada”. **NIL atau Null ≠ 0**.

Berikut adalah contoh program *linked list* yang mengimplementasikan *pointer*.

```
program linkedList;
uses crt;

type
    ptr = ^data;
    data = Record
        nilai : integer;
        next  : pointer;
    end;

var list    : ptr;
    pilihan : char;
    angka   : integer;

procedure buatNode(var paramList: ptr; paramAngka: integer);
var nodeBaru : ptr;
begin
    new(nodeBaru);
    nodeBaru^.nilai := paramAngka;
    nodeBaru^.next  := nil;

    if (paramList = nil) then paramList := nodeBaru
    else begin
        nodeBaru^.next := paramList;
        list          := nodeBaru;
    end
end;
end;
```

```
begin
    clrscr;
    new(list);
    list := nil;
    pilihan := 'y';

    while (upcase(pilihan) = 'Y') do
begin
    write('Masukkan angka: '); readln(angka);
    buatNode(list, angka);
    writeln();
    write('Tambah node baru? [y/t]: '); readln(pilihan);
end;

write('Linked List: ');

while (list <> nil) do
begin
    write(list^.nilai, ', ');
    list := list^.next;
end;
end.
```

IV. Tugas

1. Buatlah sebuah program yang meng-*input*-kan nama dan harga makanan, lalu dengan menggunakan *variable pointer* carilah:
 - a. Harga termahal serta nama harga makanan termahal.
 - b. Harga termurah serta nama harga makanan termurah.
2. Buatlah sebuah program yang menunjukkan bahwa satu buah variabel dapat ditunjuk oleh dua buah *pointer* yang berbeda.
3. Buatlah sebuah program untuk menampilkan alamat dari 5 variabel biasa yang berbeda.