PADA ERA DIGITAL SAAT INI, KEMAMPUAN PEMROGRAMAN KOMPUTER MENJADI SALAH SATU KOMPETENSI PENTING YANG HARUS DIMILIKI OLEH MAHASISWA, KHUSUSNYA MAHASISWA PENDIDIKAN FISIKA. KEMAMPUAN INI MENJADI DASAR UNTUK MENGEMBANGKAN BERBAGAI APLIKASI DAN SISTEM YANG DAPAT DIGUNAKAN DALAM BERBAGAI BIDANG, TERMASUK BIDANG FISIKA. BUKU INI DISUSUN DENGAN TUJUAN UNTUK MEMBERIKAN PEMAHAMAN DASAR TENTANG KONSEP-KONSEP PEMROGRAMAN KOMPUTER MENGGUNAKAN BAHASA PASCAL. BAHASA PASCAL DIPILIH SEBAGAI BAHASA PEMROGRAMAN DASAR KARENA MUDAH DIPELAJARI DAN MEMILIKI STRUKTUR YANG TERSTRUKTUR DAN LOGIS. BAHASA PASCAL MERUPAKAN SALAH SATU BAHASA PEMROGRAMAN YANG POPULER DIGUNAKAN UNTUK MENGEMBANGKAN APLIKASI DAN SISTEM. DICIPTAKAN OLEH NIKLAUS WIRTH PADA TAHUN 1970, PASCAL MENAWARKAN SINTAKS YANG MUDAH DIPAHAMI DAN MENGEDEPANKAN KONSEP PEMROGRAMAN STRUKTURAL. HAL INI MENJADIKAN PASCAL SEBAGAI BAHASA PEMROGRAMAN YANG TEPAT UNTUK DIGUNAKAN SEBAGAI BAHASA PEMROGRAMAN AWAL BAGI PEMBACA.

ISBN 978-623-89009-8-5

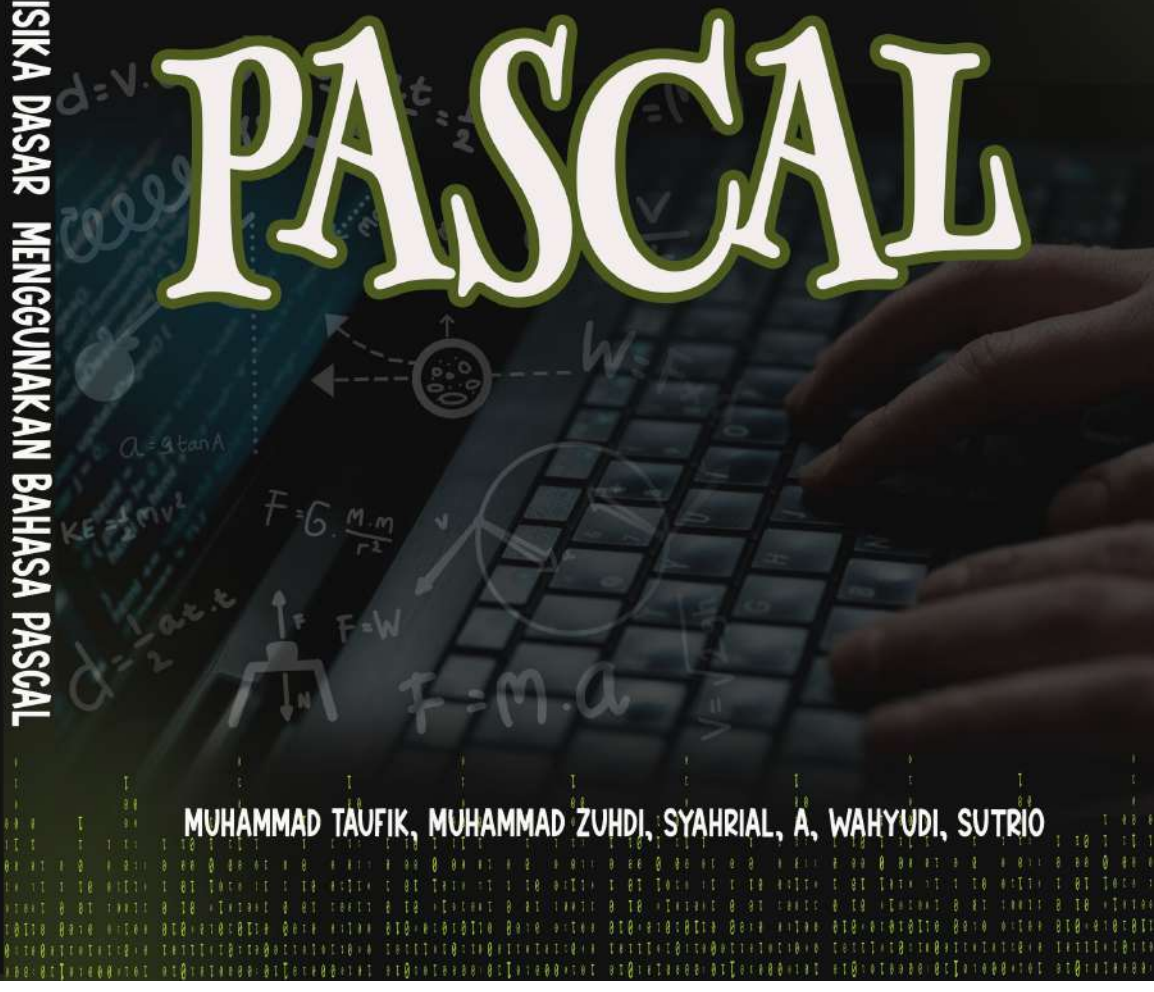
Penerbit Einstein College
Ruko BKA II/4 Jl. Gadjah Mada Jempong Baru Mataram 83116

https://lembagaeinsteincollege.com
+6281803711284
admin@lembagaeinsteincollege.com

PEMROGRAMAN FISIKA DASAR MENGGUNAKAN BAHASA PASCAL

ISBN : 978-623-89009-8-5

PEMROGRAMAN FISIKA DASAR MENGGUNAKAN BAHASA PASCAL



MUHAMMAD TAUFIK, MUHAMMAD ZUHDI, SYAHRIAL, A, WAHYUDI, SUTRIO

Muhammad Taufik,
Muhammad Zuhdi,
Syahrial, A,
Wahyudi,
Sutrio

Pemrograman Fisika Dasar Menggunakan Bahasa Pascal

PENERBIT EINSTEIN COLLEGE
ikatlah ilmu dengan tulisan

Pemrograman Fisika Dasar Menggunakan Bahasa Pascal
Penulis : Muhammad Taufik, Muhammad Zuhdi, Syahrial, A, Wahyudi, Sutrio
Editor dan Desain Cover : Lalu Muh Agri Yusna, S.Pd
Diterbitkan oleh : **Einstein College**
Ruko BKA II/4, Jl. Gadjah Mada Jempong Baru Mataram NTB 83116,
Wa : +6281803711284
Email : admin@lembagaeinsteincollege.com
Web : https://lembagaeinsteincollege.com/
Tahun Cetak : Juni, 2025

ISBN : 978-623-89009-8-5
IKAPI : 019/NTB/20023

Hak cipta dilindungi Undang-undang

Dilarang mencetak atau memperbanyak sebagian atau seluruh isi buku dalambentuk dan cara apapun tanpa ijin tertulis dari Penulis dan Penerbit.

PENERBIT EINSTEIN COLLEGE
ikatlah ilmu dengan tulisan

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan buku "Pemrograman Fisika Dasar Menggunakan Bahasa Pascal" ini. Buku ini disusun sebagai panduan bagi mahasiswa S1 Semester 5 Pendidikan Fisika PMIPA FKIP Universitas Mataram dalam mempelajari mata kuliah Pemrograman Komputer.

Pada era digital saat ini, kemampuan pemrograman komputer menjadi salah satu kompetensi penting yang harus dimiliki oleh pembaca. Kemampuan ini menjadi dasar untuk mengembangkan berbagai aplikasi dan sistem yang dapat digunakan dalam berbagai bidang, termasuk bidang fisika.

Buku ini disusun dengan tujuan untuk memberikan pemahaman dasar tentang konsep-konsep pemrograman komputer menggunakan bahasa Pascal. Bahasa Pascal dipilih sebagai bahasa pemrograman dasar karena mudah dipelajari dan memiliki struktur yang terstruktur dan logis.

Materi dalam buku ini disusun secara sistematis dan mudah dipahami, dengan dilengkapi contoh-contoh dan latihan soal untuk membantu mahasiswa dalam memahami konsep-konsep yang dipelajari.

Buku ini terdiri dari 12 bab yang membahas berbagai topik penting dalam pemrograman komputer, antara lain:Pendahuluan Pemrograman Pascal, Variabel dan Tipe Data, Operasi, Aritmatika dan Logika, Percabangan, Perulangan, Fungsi dan Prosedur, Array, String, Rekursif, File Handling, Struktur Data Lanjutan, Proyek Akhir

Penulis berharap buku ini dapat bermanfaat bagi bagi pembaca dalam mempelajari Pemrograman Komputer. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyusunan buku ini.

Mataram, Juni 2025

Penulis

DAFTAR ISI

BAB I
PENGENALAN PEMROGRAMAN PASCAL DENGAN FREE PASCAL

A. Konsep Dasar Pemrograman Pascal

Pemrograman Pascal terdiri dari beberapa elemen dasar, yaitu:

1. Variabel: Variabel adalah tempat untuk menyimpan data yang akan digunakan dalam program. Variabel memiliki nama dan tipe data yang menentukan jenis data yang dapat disimpan di dalamnya.

2. Tipe data: Tipe data adalah jenis data yang dapat disimpan dalam variabel. Tipe data yang umum digunakan dalam Pascal adalah integer, real, char, dan string.
3. Operator: Operator adalah simbol yang digunakan untuk melakukan operasi terhadap data. Operator yang umum digunakan dalam Pascal adalah operator aritmatika, operator relasional, dan operator logika.
4. Struktur kontrol: Struktur kontrol adalah pernyataan yang digunakan untuk mengontrol alur program. Struktur kontrol yang umum digunakan dalam Pascal adalah if-then-else, while, dan for.

B. Menginstall Free Pascal

Free Pascal dapat diunduh secara gratis dari situs web resmi https://www.freepascal.org/download.html. Setelah diunduh, ikuti petunjuk instalasi yang disediakan untuk menginstal Free Pascal di komputer Anda.

C. Menulis Program Pascal Sederhana

Berikut adalah contoh program Pascal sederhana yang menjumlahkan dua bilangan:

```
program MenjumlahkanDuaBilangan;

var
  x: integer;
  y: integer;
  z: integer;

begin
  Write('Masukkan bilangan pertama: ');
```

```
  ReadLn(x);
  Write('Masukkan bilangan kedua: ');
  ReadLn(y);

  z := x + y;

  WriteLn('Hasil penjumlahan: ', z);
End.
```

Program di atas pertama-tama mendeklarasikan tiga variabel, yaitu x, y, dan z. Variabel x dan y bertipe data integer dan digunakan untuk menyimpan nilai bilangan yang diinputkan oleh pengguna. Variabel z juga bertipe data integer dan digunakan untuk menyimpan hasil penjumlahan.

Selanjutnya, program menggunakan prosedur WriteLn untuk menampilkan pesan ke layar dan prosedur ReadLn untuk membaca nilai yang diinputkan oleh pengguna. Nilai yang diinputkan disimpan dalam variabel x dan y. Setelah itu, program melakukan operasi penjumlahan x + y dan menyimpan hasilnya dalam variabel z. Terakhir, program menggunakan prosedur WriteLn untuk menampilkan hasil penjumlahan ke layar.

Berikut adalah beberapa contoh soal dan latihan dengan program Pascal untuk perhitungan fisika:

Soal 1: Menghitung Luas Persegi Panjang

Penjelasan:

Luas persegi panjang dihitung dengan rumus luas = panjang * lebar.

Program Pascal:

```
program MenghitungLuasPersegiPanjang;

var
  panjang: real;
  lebar: real;
  luas: real;

begin
```

```
  Write('Masukkan panjang persegi panjang: ');
  ReadLn(panjang);
  Write('Masukkan lebar persegi panjang: ');
  ReadLn(lebar);

  luas := panjang * lebar;

  WriteLn('Luas persegi panjang: ', luas);
end.
```

Penjelasan program:

1. Program mendeklarasikan variabel panjang, lebar, dan luas dengan tipe data real.
2. Program meminta pengguna untuk memasukkan nilai panjang dan lebar persegi panjang.
3. Program menghitung luas persegi panjang dengan mengalikan panjang dan lebar.
4. Program menampilkan hasil luas persegi panjang ke layar.

Soal 2: Menghitung Keliling Lingkaran

Penjelasan:

Keliling lingkaran dihitung dengan rumus keliling = 2 * π * jari-jari.

Program Pascal:

```
program MenghitungKelilingLingkaran;

const
  pi = 3.14159265358979323846;

var
  jariJari: real;
  keliling: real;

begin
  Write('Masukkan jari-jari lingkaran: ');
  ReadLn(jariJari);
```

```
  keliling := 2 * pi * jariJari;

  WriteLn('Keliling lingkaran: ', keliling);
end.
```

Penjelasan program:

1. Program mendeklarasikan variabel jariJari dan keliling dengan tipe data real.
2. Program mendefinisikan konstanta pi dengan nilai 3.14159265358979323846.
3. Program meminta pengguna untuk memasukkan nilai jari-jari lingkaran.
4. Program menghitung keliling lingkaran dengan mengalikan 2 * pi dan jariJari.
5. Program menampilkan hasil keliling lingkaran ke layar.

Soal 3: Mengonversi Suhu dari Celsius ke Fahrenheit

Penjelasan:

Rumus untuk mengonversi suhu dari Celsius ke Fahrenheit adalah Fahrenheit = (Celsius * 9/5) + 32.

Program Pascal:

```
program MengonversiSuhuCelsiusKeFahrenheit;

var
  celsius: real;
  fahrenheit: real;

begin
  Write('Masukkan suhu Celsius: ');
  ReadLn(celsius);

  fahrenheit := (celsius * 9/5) + 32;

  WriteLn('Suhu Fahrenheit: ', fahrenheit);
end.
```

Penjelasan program:

1. Program mendeklarasikan variabel Celsius dan fahrenheit dengan tipe data real.
2. Program meminta pengguna untuk memasukkan nilai suhu Celsius.
3. Program menghitung suhu Fahrenheit dengan mengalikan Celsius dengan 9/5 dan menambahkan 32.
4. Program menampilkan hasil suhu Fahrenheit ke layar.

Latihan:

Tulis program Pascal sederhana untuk menyelesaikan berbagai masalah perhitungan fisika lainnya, seperti menghitung gaya gravitasi, menghitung momentum, atau menghitung energi kinetik.

D. Rangkuman

Pada bab ini, Anda telah mempelajari konsep dasar pemrograman Pascal, termasuk struktur program Pascal, variabel, tipe data, operator, dan struktur kontrol. Anda juga telah belajar cara menginstal dan menggunakan compiler Free Pascal, serta menulis program Pascal sederhana untuk menyelesaikan berbagai masalah.

E. Test Formatif

1. Jelaskan apa yang dimaksud dengan variabel dalam pemrograman Pascal.
2. Sebutkan beberapa tipe data yang umum digunakan dalam Pascal.
3. Jelaskan fungsi operator aritmatika dalam Pascal.
4. Berikan contoh struktur kontrol if-then-else dalam Pascal.
5. Jelaskan langkah-langkah untuk menginstal Free Pascal di komputer Anda.

F. Umpan Balik

Setelah menyelesaikan bab ini, diharapkan Anda dapat memahami dan menerapkan konsep-konsep dasar pemrograman Pascal. Jika Anda masih memiliki pertanyaan atau kesulitan, jangan ragu untuk bertanya kepada dosen atau pengajar.

G. Tindak Lanjut

1. Pelajari lebih lanjut tentang struktur kontrol yang lebih kompleks dalam Pascal.
2. Cobalah untuk menulis program Pascal yang lebih kompleks untuk menyelesaikan berbagai masalah.
3. Ikuti pelatihan atau kursus pemrograman Pascal untuk meningkatkan kemampuan Anda.

H. Kunci Jawaban Formatif

1. Variabel adalah tempat untuk menyimpan data yang akan digunakan dalam program. Variabel memiliki nama dan tipe data yang menentukan jenis data yang dapat disimpan di dalamnya.
2. Beberapa tipe data yang umum digunakan dalam Pascal adalah integer, real, char, dan string.
3. Operator aritmatika digunakan untuk melakukan operasi aritmatika terhadap data, seperti penjumlahan, pengurangan, perkalian, dan pembagian.
4. Berikut adalah contoh struktur kontrol if-then-else dalam Pascal:

```
if kondisi then
  pernyataan1
else
  pernyataan2
```

5. Langkah-langkah untuk menginstal Free Pascal di komputer Anda:

a. Unduh installer Free Pascal dari situs web resmi https://www.freepascal.org/download.html.
b. Jalankan installer dan ikuti petunjuk di layar.
c. Pilih compiler dan IDE yang ingin Anda instal.
d. Klik tombol "Install" untuk menyelesaikan proses instalasi.

BAB II.
VARIABEL DAN TIPE DATA

A. Pengertian Variabel

Variabel adalah elemen fundamental dalam pemrograman yang digunakan untuk menyimpan nilai. Nilai ini dapat berupa angka, karakter, string, atau data lainnya. Setiap variabel memiliki nama yang unik dan tipe data yang menentukan jenis nilai yang dapat disimpannya.

B. Tipe Data

Tipe data mendefinisikan jenis nilai yang dapat disimpan dalam variabel. Tipe data yang umum digunakan dalam bahasa pemrograman Pascal antara lain:

a. Integer: Digunakan untuk menyimpan nilai bilangan bulat, seperti 1, -10, 2000.
b. Real: Digunakan untuk menyimpan nilai bilangan desimal, seperti 3.14, -5.2, 100.000.
c. Char: Digunakan untuk menyimpan satu karakter, seperti 'a', 'Z', ''.
d. String: Digunakan untuk menyimpan urutan karakter, seperti "Halo", "Dunia!", "Pemrograman Pascal".

C. Deklarasi Variabel

Deklarasi variabel adalah proses untuk memberitahu kompilator tentang nama dan tipe data variabel yang akan digunakan dalam program. Deklarasi variabel dilakukan dengan menggunakan kata kunci `var` diikuti dengan nama variabel, tipe data, dan tanda titik koma. Berikut contoh deklarasi variabel:

```
var
  x, y: integer;
  z: real;
  nama: string(20);
```

Pada contoh diatas, variabel `x` dan `y` bertipe data `integer`, variabel `z` bertipe data `real`, dan variabel `nama` bertipe data `string` dengan panjang maksimum 20 karakter.

D. Inisialisasi Variabel

Inisialisasi variabel adalah proses untuk memberikan nilai awal kepada variabel saat dideklarasikan. Inisialisasi variabel dilakukan dengan menggunakan tanda sama dengan (`=`) diikuti dengan nilai yang ingin diberikan. Berikut contoh inisialisasi variabel:

```
var
  x: integer = 10;
  y: real = 3.14;
  z: char = 'A';
  nama: string(20) = 'Belajar Pascal';
```

Pada contoh diatas, variabel `x` diinisialisasi dengan nilai 10, variabel `y` diinisialisasi dengan nilai 3.14, variabel `z` diinisialisasi dengan karakter 'A', dan variabel `nama` diinisialisasi dengan string "Belajar Pascal".

E. Penggunaan Variabel

Variabel digunakan dalam program untuk menyimpan nilai, melakukan operasi, dan menghasilkan keluaran. Berikut contoh penggunaan variabel:

```
var
  x: integer = 10;
  y: real = 5;

begin
  z := x + y;
  WriteLn('Hasil penjumlahan: ', z);
end.
```

Pada contoh diatas, variabel `x` dan `y` digunakan untuk menyimpan nilai 10 dan 5. Nilai `x` dan `y` kemudian

```
var
  x, y: integer;
  z: real;
  nama: string(20);
```

Pada contoh diatas, variabel x dan y bertipe data integer, variabel z bertipe data real, dan variabel nama bertipe data string dengan panjang maksimum 20 karakter.

F. Inisialisasi Variabel

Inisialisasi variabel adalah proses untuk memberikan nilai awal kepada variabel saat dideklarasikan. Inisialisasi variabel dilakukan dengan menggunakan tanda sama dengan (=) diikuti dengan nilai yang ingin diberikan. Berikut contoh inisialisasi variabel:

```
var
  x: integer = 10;
  y: real = 3.14;
  z: char = 'A';
  nama: string(20) = 'Belajar Pascal';
```

Pada contoh diatas, variabel x diinisialisasi dengan nilai 10, variabel y diinisialisasi dengan nilai 3.14, variabel z diinisialisasi dengan karakter 'A', dan variabel nama diinisialisasi dengan string "Belajar Pascal".

G. Penggunaan Variabel

Variabel digunakan dalam program untuk menyimpan nilai, melakukan operasi, dan menghasilkan keluaran. Berikut contoh penggunaan variabel:

```
var
  x: integer = 10;
  y: real = 5;

begin
  z := x + y;
  WriteLn('Hasil penjumlahan: ', z);
end.
```

H. Konversi Tipe Data

Konversi tipe data adalah proses mengubah nilai dari satu tipe data ke tipe data lain. Konversi tipe data dapat dilakukan secara implisit atau eksplisit.

a. Konversi Implisit: Dilakukan secara otomatis oleh kompilator. Contohnya, ketika nilai integer dijumlahkan dengan nilai real, maka nilai integer akan dikonversi secara implisit ke nilai real sebelum operasi penjumlahan dilakukan.

b. Konversi Eksplisit: Dilakukan secara manual oleh programmer dengan menggunakan fungsi atau operator konversi tipe data. Contohnya, menggunakan fungsi StrToInt untuk mengonversi string menjadi integer.

I. Kesalahan Umum dalam Penggunaan Variabel

Kesalahan umum dalam penggunaan variabel antara lain:

a. Deklarasi variabel yang tidak tepat: Tidak mendeklarasikan variabel sebelum menggunakannya atau mendeklarasikan variabel dengan tipe data yang salah.
b. Inisialisasi variabel yang tidak lengkap: Tidak memberikan nilai awal kepada variabel saat dideklarasikan.
c. Penggunaan variabel yang tidak diinisialisasi: Menggunakan variabel yang belum diinisialisasi untuk melakukan operasi.
d. Penggunaan variabel dengan tipe data yang tidak sesuai: Menggunakan variabel dengan tipe data yang tidak sesuai dengan operasi yang dilakukan.

J. Tips Penggunaan Variabel yang Baik

Tips penggunaan variabel yang baik antara lain:

a. Gunakan nama variabel yang deskriptif dan mudah dipahami.
b. Pilih tipe data yang tepat untuk variabel sesuai dengan kebutuhan program.
c. Inisialisasi variabel dengan nilai awal yang sesuai.
d. Periksa nilai variabel sebelum menggunakannya dalam operasi.
e. Hindari penggunaan variabel yang tidak diinisialisasi.
f. Gunakan komentar untuk menjelaskan penggunaan variabel.

Berikut ini disajikan beberapa contoh program yang dikaitkan dengan fisika, yaitu program kecepatan rata-rata, program energi kinetik, program gaya gravitasi, momentum, kerja mekanik, program gerak lurus berubah setiap saat, dan program gerak melingkar berubah beraturan.

```
program KecepatanRataRata;

var
  jarak: real;
  waktu: real;
  kecepatan: real;
```

```
begin
  Write('Masukkan jarak (dalam meter): ');
  ReadLn(jarak);
  Write('Masukkan waktu (dalam detik): ');
  ReadLn(waktu);

  kecepatan := jarak / waktu;

  WriteLn('Kecepatan rata-rata: ', kecepatan:4:2, ' m/s');
end.
```

```
program EnergiKinetik;

var
  massa: real;
  kecepatan: real;
  energiKinetik: real;

begin
  Write('Masukkan massa (dalam kg): ');
  ReadLn(massa);
  Write('Masukkan kecepatan (dalam m/s): ');
  ReadLn(kecepatan);

  energiKinetik := 0.5 * massa * kecepatan * kecepatan;

  WriteLn('Energi kinetik: ', energiKinetik:4:2, ' J');
end.
```

```
program GayaGravitasi;

var
```

```
  massa1: real;
  massa2: real;
  jarak: real;
  gayaGravitasi: real;

const
  G = 6.67430e-11; { Konstanta gravitasi umum (m^3 kg^-1 s^-2) }

begin
  Write('Masukkan massa benda pertama (dalam kg): ');
  ReadLn(massa1);
  Write('Masukkan massa benda kedua (dalam kg): ');
  ReadLn(massa2);
  Write('Masukkan jarak antar benda (dalam meter): ');
  ReadLn(jarak);

  gayaGravitasi := G * massa1 * massa2 / (jarak * jarak);

  WriteLn('Gaya gravitasi: ', gayaGravitasi:4:2, ' N');
end.
```

```
program Momentum;

var
  massa: real;
  kecepatan: real;
  momentum: real;

begin
  Write('Masukkan massa (dalam kg): ');
  ReadLn(massa);
  Write('Masukkan kecepatan (dalam m/s): ');
  ReadLn(kecepatan);
```

```
  momentum := massa * kecepatan;

  WriteLn('Momentum: ', momentum:4:2, ' kg m/s');
end.
```

```
program KerjaMekanik;

var
  gaya: real;
  perpindahan: real;
  kerjaMekanik: real;

begin
  Write('Masukkan gaya (dalam N): ');
  ReadLn(gaya);
  Write('Masukkan perpindahan (dalam meter): ');
  ReadLn(perpindahan);

  kerjaMekanik := gaya * perpindahan;

  WriteLn('Kerja mekanik: ', kerjaMekanik:4:2, ' J');
end.
```

```
program GerakLurusBerubahSewaktu;

var
  waktuAwal: real;
  waktuAkhir: real;
  kecepatanAwal: real;
  kecepatanAkhir: real;
  percepatan: real;
  jarak: real;

begin
```

```
  Write('Masukkan waktu awal (dalam detik): ');
  ReadLn(waktuAwal);
  Write('Masukkan waktu akhir (dalam detik): ');
  ReadLn(waktuAkhir);
  Write('Masukkan kecepatan awal (dalam m/s): ');
  ReadLn(kecepatanAwal);
  Write('Masukkan kecepatan akhir (dalam m/s): ');
  ReadLn(kecepatanAkhir);

  percepatan := (kecepatanAkhir - kecepatanAwal) / (waktuAkhir -
waktuAwal);
  jarak := 0.5 * (kecepatanAwal + kecepatanAkhir) * (waktuAkhir -
waktuAwal);

  WriteLn('Percepatan: ', percepatan:4:2, ' m/s^2');
  WriteLn('Jarak: ', jarak:4:2, ' m');
end.
```

```
program GerakMelingkarBerubahBeraturan;

var
  sudutAwal: real;
  sudutAkhir: real;
  kecepatanSudutAwal: real;
  kecepatanSudutAkhir: real;
  percepatanSudut: real;
  jarakSudut: real;
  kecepatanLinearAwal: real;
  kecepatanLinearAkhir: real;
  jariJari: real;

begin
  Write('Masukkan sudut awal (dalam radian): ');
  ReadLn(sudutAwal);
```

```
  Write('Masukkan sudut akhir (dalam radian): ');
  ReadLn(sudutAkhir);
  Write('Masukkan kecepatan sudut awal (dalam rad/s): ');
  ReadLn(kecepatanSudutAwal);
  Write('Masukkan kecepatan sudut akhir (dalam rad/s): ');
  ReadLn(kecepatanSudutAkhir);

  percepatanSudut := (kecepatanSudutAkhir - kecepatanSudutAwal) /
(sudutAkhir - sudutAwal);
  jarakSudut := 0.5 * (kecepatanSudutAwal + kecepatanSudutAkhir) *
(sudutAkhir - sudutAwal);
  kecepatanLinearAwal := kecepatanSudutAwal * jariJari;
  kecepatanLinearAkhir := kecepatanSudutAkhir * jariJari;

  WriteLn('Percepatan sudut: ', percepatanSudut:4:2, ' rad/s^2');
  WriteLn('Jarak sudut: ', jarakSudut:4:2, ' rad');
  WriteLn('Kecepatan linear awal: ', kecepatanLinearAwal:4:2, ' m/s');
  WriteLn('Kecepatan linear akhir: ', kecepatanLinearAkhir:4:2, ' m/s');
end.
```

K. Rangkuman:

Variabel dan tipe data merupakan konsep fundamental dalam pemrograman Pascal. Memahami variabel dan tipe data dengan baik akan membantu Anda dalam menulis program yang lebih terstruktur, efisien, dan mudah dipahami.

L. Test Formatif

1. Jelaskan apa yang dimaksud dengan variabel dan tipe data dalam bahasa pemrograman Pascal.
2. Sebutkan beberapa tipe data yang umum digunakan dalam bahasa pemrograman Pascal.
3. Bagaimana cara mendeklarasikan dan menginisialisasi variabel dalam bahasa pemrograman Pascal?
4. Jelaskan apa yang dimaksud dengan konversi tipe data dan berikan contohnya.
5. Sebutkan beberapa kesalahan umum dalam penggunaan variabel dan jelaskan cara menghindarinya.

M.Umpan Balik

Setelah mempelajari bab ini, diharapkan Anda dapat memahami konsep variabel dan tipe data dengan baik dan dapat menggunakannya dengan benar dalam program Pascal Anda.

N. Tindak Lanjut

Untuk menguji pemahaman dan mengaplikasikan materi, kerjakan latihan, diskusikan dengan dosen atau teman sekelas, dan buatlah program Pascal sederhana yang menggunakan variabel dan tipe data.

O. Kunci Jawaban Formatif

1. Variabel: Elemen fundamental dalam pemrograman yang digunakan untuk menyimpan nilai. Tipe data: Menentukan jenis nilai yang dapat disimpan dalam variabel.
2. Integer, real, char, string.
3. Deklarasi: var nama_variabel: tipe_data;. Inisialisasi: nama_variabel := nilai_awal;.
4. Konversi tipe data adalah proses mengubah nilai dari satu tipe data ke tipe data lain. Contoh: StrToInt(string) untuk mengonversi string menjadi integer.
5. Kesalahan umum:
   a. Deklarasi tidak tepat.
   b. Inisialisasi tidak lengkap.
   c. Penggunaan variabel tidak diinisialisasi.
   d. Penggunaan tipe data tidak sesuai. Tips:
   e. Gunakan nama variabel deskriptif.
   f. Pilih tipe data tepat.
   g. Inisialisasi variabel.
   h. Periksa nilai variabel.
   i. Hindari penggunaan variabel tidak diinisialisasi.
   j. Gunakan komentar.

BAB III.
OPERASI ARITMATIKA DAN LOGIKA

A. Operasi Aritmatika

Operasi aritmatika digunakan untuk melakukan perhitungan matematis terhadap data numerik. Operasi aritmatika yang umum digunakan dalam Pascal adalah: Penjumlahan (+), Pengurangan (-), Perkalian (*), Pembagian (/), Modulo (%), Pangkat (^ atau)

Contoh penggunaan operasi aritmatika dalam Pascal untuk menghitung luas perseg ipanjang. Berikut sajian program nya.

```
program MenghitungLuasPersegiPanjang;

var
  panjang: real;
  lebar: real;
  luas: real;

begin
  panjang := 10;
  lebar := 5;

  luas := panjang * lebar;

  WriteLn('Luas persegi panjang: ', luas);
end.
```

Pada contoh program di atas, operasi aritmatika `*` digunakan untuk menghitung luas persegi panjang.

B. Operasi Logika

Operasi logika digunakan untuk membuat keputusan dalam program berdasarkan kondisi tertentu. Operasi logika yang umum digunakan dalam Pascal adalah: NOT (¬), AND (∧) dan OR (∨).

Contoh penggunaan operasi logika dalam Pascal seperti contoh 01 untuk memeriksa nilai ujian

Contoh 01.

```
program MemeriksaNilaiUjian;
```

```
var
  nilaiUjian: integer;
  lulus: boolean;
begin
  Write('Masukkan nilai ujian: ');
  ReadLn(nilaiUjian);

  lulus := (nilaiUjian >= 70);

  if lulus then
    WriteLn('Selamat, Anda lulus!')
  else
    WriteLn('Maaf, Anda tidak lulus.');
end.
```

Pada contoh program 01, operasi logika `>=` digunakan untuk memeriksa apakah nilai ujian lebih dari atau sama dengan 70. Jika kondisi terpenuhi, program akan menampilkan pesan "Selamat, Anda lulus!". Jika kondisi tidak terpenuhi, program akan menampilkan pesan "Maaf, Anda tidak lulus.".

Selanjutnya disajikan contoh Program Pascal untuk Perhitungan Fisika disajikan pada contoh 02,03,04 dan 05.

Contoh 02. Menghitung Jarak Bayangan Cermin Datar

Penjelasan:

Jarak bayangan pada cermin datar sama dengan jarak benda ke cermin. Rumus untuk menghitung jarak bayangan pada cermin datar adalah `jarak_bayangan = jarak_benda`.

```
program MenghitungJarakBayanganCerminDatar;

var
  jarakBenda: real;
  jarakBayangan: real;

begin
  Write('Masukkan jarak benda ke cermin: ');
  ReadLn(jarakBenda);
```

```
  jarakBayangan := jarakBenda;

  WriteLn('Jarak bayangan pada cermin datar: ', jarakBayangan);
end.
```

Contoh 03. Menghitung Sudut Bias pada Refraksi Cahaya

Penjelasan:

Sudut bias pada refraksi cahaya dapat dihitung dengan rumus `sin(sudut_bias) = (n2/n1) * sin(sudut_datang)`, di mana:

a. `n1` adalah indeks bias medium pertama
b. `n2` adalah indeks bias medium kedua
c. `sudut_datang` adalah sudut datang cahaya
d. `sudut_bias` adalah sudut bias cahaya

```
program MenghitungSudutBiasRefraksi;

const
  pi = 3.14159265358979323846;

var
  n1: real;
  n2: real;
  sudutDatang: real;
  sudutBias: real;

begin
  Write('Masukkan indeks bias medium pertama: ');
  ReadLn(n1);
  Write('Masukkan indeks bias medium kedua: ');
  ReadLn(n2);
  Write('Masukkan sudut datang cahaya (dalam derajat): ');
  ReadLn(sudutDatang);

  sudutDatang := sudutDatang * pi / 180;
  sudutBias := asin((n2/n1) * sin(sudutDatang));
  sudutBias := sudutBias * 180 / pi;
```

```
  WriteLn('Sudut bias cahaya (dalam derajat): ', sudutBias);
end.
```

Contoh 04. Menghitung Panjang Gelombang Cahaya

Penjelasan:

Panjang gelombang cahaya dapat dihitung dengan rumus `lambda = v/f`, di mana:

a. `lambda` adalah panjang gelombang cahaya
b. `v` adalah kecepatan cahaya dalam medium
c. `f` adalah frekuensi cahaya

```
program MenghitungPanjangGelombangCahaya;

const
  c = 299792458; // Kecepatan cahaya dalam ruang hampa (m/s)

var
  medium: string;
  indeksBias: real;
  frekuensi: real;
  panjangGelombang: real;

begin
  Write('Masukkan medium cahaya (ruang hampa, air, kaca): ');
  ReadLn(medium);

  if medium = 'ruang hampa' then
    indeksBias := 1
  else if medium = 'air' then
    indeksBias := 1.33
  else if medium = 'kaca' then
    indeksBias := 1.5
  else
    begin
      WriteLn('Medium tidak valid.');
      Exit;
```

```
  end;

  Write('Masukkan frekuensi cahaya (Hz): ');
  ReadLn(frekuensi);

  panjangGelombang := c / (frekuensi * indeksBias);

  WriteLn('Panjang gelombang cahaya (m): ', panjangGelombang);
end.
```

Contoh 05. Menghitung Intensitas Cahaya

Penjelasan:

Intensitas cahaya dapat dihitung dengan rumus `I = P/A`, di mana:

a. `I` adalah intensitas cahaya (W/m²)
b. `P` adalah daya cahaya (W)
c. `A` adalah luas area yang diiluminasi (m²)

```
program MenghitungIntensitasCahaya;

var
  dayaCahaya: real;
  luasArea: real;
  intensitasCahaya: real;

begin
  Write('Masukkan daya cahaya (W): ');
  ReadLn(dayaCahaya);
  Write('Masukkan luas area yang diiluminasi (m²): ');
  ReadLn(luasArea);

  intensitasCahaya := dayaCahaya / luasArea;

  WriteLn('Intensitas cahaya (W/m²): ', intensitasCahaya);
end.
```

Contoh 06. Menghitung Energi Foton

Penjelasan:

Energi foton dapat dihitung dengan rumus `E = hf`, di mana:

a. `E` adalah energi foton (J)
b. `h` adalah konstanta Planck (6.62607004 × 10^-34 J·s)
c. `f` adalah frekuensi cahaya (Hz)

```
program MenghitungEnergiFoton;

const
  h = 6.62607004e-34; // Konstanta Planck (J·s)

var
  frekuensi: real;
  energiFoton: real;

begin
  Write('Masukkan frekuensi cahaya (Hz): ');
  ReadLn(frekuensi);

  energiFoton := h * frekuensi;

  WriteLn('Energi foton (J): ', energiFoton);
end.
```

C. Rangkuman

Pada bab ini, Anda telah mempelajari operasi aritmatika dan logika dalam pemrograman Pascal. Operasi aritmatika digunakan untuk melakukan perhitungan matematis, sedangkan operasi logika digunakan untuk membuat keputusan dalam program. Anda juga telah belajar cara menggunakan operasi-operasi ini untuk menyelesaikan berbagai masalah.

D. Test Formatif

1. Jelaskan apa yang dimaksud dengan operasi aritmatika.
2. Sebutkan beberapa operasi aritmatika yang umum digunakan dalam Pascal.
2. Jelaskan fungsi operasi logika.

3. Berikan contoh penggunaan operasi logika `AND` dalam Pascal.
4. Tuliskan program Pascal yang menghitung rata-rata tiga bilangan.

E. Umpan Balik

Setelah menyelesaikan bab ini, diharapkan Anda dapat memahami dan menerapkan operasi aritmatika dan logika dalam pemrograman Pascal. Jika Anda masih memiliki pertanyaan atau kesulitan, jangan ragu untuk bertanya kepada dosen atau pengajar.

F. Tindak Lanjut

Setelah memahami operasi aritmatika dan logika dasar, lanjutkan dengan mempelajari operasi yang lebih kompleks untuk memperluas kemampuan Anda dalam menyelesaikan permasalahan komputasi. Cobalah menulis program Pascal yang lebih kompleks yang memanfaatkan operasi aritmatika dan logika secara kreatif.

G. Kunci Jawaban Formatif

1. Operasi aritmatika adalah operasi matematika dasar yang digunakan untuk melakukan perhitungan terhadap dua atau lebih bilangan. Operasi aritmatika yang umum digunakan dalam pemrograman Pascal meliputi:
   - Penjumlahan (+)
   - Pengurangan (-)
   - Perkalian (*)
   - Pembagian (/)
   - Pembagian modulo (%)
   - Pangkat (^ atau **)
2. Operasi aritmatika dalam Pascal dapat digunakan untuk menghitung nilai numerik, variabel, dan ekspresi. Urutan operasi aritmatika dalam Pascal mengikuti aturan matematika standar, yaitu:
   1. Pangkat
   2. Perkalian dan pembagian (dari kiri ke kanan)
   3. Penjumlahan dan pengurangan (dari kiri ke kanan)
3. Operasi logika digunakan untuk membuat keputusan dalam program berdasarkan kondisi tertentu. Operasi logika memungkinkan programmer untuk menentukan apakah suatu pernyataan benar atau salah. Operasi logika yang umum digunakan dalam Pascal meliputi:
   - NOT (¬)
   - AND (∧)
   - OR (∨)

4. Contoh Penggunaan Operasi Logika `AND` dalam Pascal

```
program RataRataTigaBilangan;

var
  bil1, bil2, bil3, rataRata: integer;

begin
  writeln('Masukkan bilangan pertama: ');
  readln(bil1);

  writeln('Masukkan bilangan kedua: ');
  readln(bil2);
  writeln('Masukkan bilangan ketiga: ');
  readln(bil3);

  rataRata := (bil1 + bil2 + bil3) / 3;

  if (bil1 > 0) and (bil2 > 0) and (bil3 > 0) then
  begin
    writeln('Rata-rata dari ', bil1, ', ', bil2, ', dan ', bil3, ' adalah ', rataRata);
  end
  else
  begin
    writeln('Ada bilangan yang tidak valid!');
  end;
end.
```

Pada contoh program di atas, operasi logika `AND` digunakan untuk memastikan bahwa semua bilangan yang dimasukkan lebih besar dari 0. Jika semua bilangan valid, program akan menghitung dan menampilkan rata-rata dari ketiga bilangan tersebut.

5. Program Pascal untuk Menghitung Rata-rata Tiga Bilangan

```
program RataRataTigaBilangan;
var
  bil1, bil2, bil3, rataRata: integer;
```

```
begin
  writeln('Masukkan bilangan pertama: ');
  readln(bil1);
    writeln('Masukkan bilangan kedua: ');
  readln(bil2);
    writeln('Masukkan bilangan ketiga: ');
  readln(bil3);
  rataRata := (bil1 + bil2 + bil3) / 3;
    writeln('Rata-rata dari ', bil1, ', ', bil2, ', dan ', bil3, ' adalah ', rataRata);
end.
```

Program ini menghitung rata-rata dari tiga bilangan yang dimasukkan oleh pengguna. Program ini menggunakan operasi aritmatika penjumlahan, pembagian, dan penugasan untuk menghitung nilai rata-rata. Hasil rata-rata kemudian ditampilkan kepada pengguna.

BAB IV.
PERCABANGAN

A. Struktur Percabangan IF-THEN-ELSE

Struktur percabangan `IF-THEN-ELSE` digunakan untuk membuat keputusan sederhana dalam program. Struktur ini terdiri dari dua bagian:

a. Bagian IF: Bagian ini mengecek kondisi yang ingin diuji. Jika kondisi terpenuhi, maka bagian THEN akan dieksekusi.
b. Bagian ELSE: Bagian ini dieksekusi jika kondisi dalam bagian IF tidak terpenuhi.

Contoh penggunaan struktur percabangan `IF-THEN-ELSE` dalam Pascal:

```
program MemeriksaNilaiUjian;

var
  nilaiUjian: integer;
  lulus: boolean;

begin
  Write('Masukkan nilai ujian: ');
  ReadLn(nilaiUjian);

  lulus := (nilaiUjian >= 70);

  if lulus then
    WriteLn('Selamat, Anda lulus!')
  else
    WriteLn('Maaf, Anda tidak lulus.');
end.
```

Pada contoh program di atas, struktur percabangan `IF-THEN-ELSE` digunakan untuk memeriksa apakah nilai ujian lebih dari atau sama dengan 70. Jika kondisi terpenuhi, program akan menampilkan pesan "Selamat, Anda lulus!". Jika kondisi tidak terpenuhi, program akan menampilkan pesan "Maaf, Anda tidak lulus.".

B. Struktur Percabangan CASE

Struktur percabangan `CASE` digunakan untuk membuat keputusan berdasarkan nilai yang dibandingkan dengan beberapa nilai yang telah ditentukan. Struktur ini terdiri dari beberapa bagian:

a. Klausa CASE: Klausa ini berisi nilai yang ingin dibandingkan.
b. Klausa OF: Klausa ini berisi daftar nilai yang dibandingkan dengan nilai dalam klausa CASE.
c. Klausa THEN: Klausa ini berisi kode yang dieksekusi jika nilai dalam klausa CASE cocok dengan salah satu nilai dalam klausa OF.
d. Klausa ELSE: Klausa ini dieksekusi jika nilai dalam klausa CASE tidak cocok dengan salah satu nilai dalam klausa OF.

Contoh penggunaan struktur percabangan `CASE` dalam Pascal:

```
program MemilihMenu;

var
  pilihan: char;

begin
  Write('Pilih menu (A, B, atau C): ');
  ReadLn(pilihan);

  case pilihan of
    'A':
      WriteLn('Anda memilih menu A.');
    'B':
      WriteLn('Anda memilih menu B.');
    'C':
      WriteLn('Anda memilih menu C.');
    else
      WriteLn('Pilihan tidak valid.');
  end;
end.
```

Pada contoh program di atas, struktur percabangan `CASE` digunakan untuk membedakan pilihan menu berdasarkan karakter yang diinputkan oleh pengguna. Jika karakter yang diinputkan cocok dengan salah satu nilai dalam klausa OF, maka klausa THEN yang sesuai akan dieksekusi. Jika karakter yang diinputkan tidak cocok dengan salah satu nilai dalam klausa OF, maka klausa ELSE akan dieksekusi.

Berikut adalah 5 contoh program Pascal untuk perhitungan fisika terkait termodinamika yang menggunakan struktur percabangan:

1. Menghitung Fase Zat Berdasarkan Tekanan dan Suhu

Penjelasan:

Fase zat (padat, cair, gas) dapat ditentukan berdasarkan tekanan dan suhu dengan menggunakan diagram fase. Program Pascal ini akan membantu menentukan fase zat berdasarkan nilai tekanan dan suhu yang dimasukkan.

Program Pascal:

```
program MenentukanFaseZat;

var
  tekanan: real;
  suhu: real;
  fase: char;

begin
  Write('Masukkan tekanan (kPa): ');
  ReadLn(tekanan);
  Write('Masukkan suhu (°C): ');
  ReadLn(suhu);

  if suhu >= 100 then
    fase := 'Gas'
  else
    if tekanan <= 101.3 then
      fase := 'Padat'
    else
      fase := 'Cair'
    end;
```

```
  end;

  WriteLn('Fase zat: ', fase);
end.
```

Penjelasan:

Program ini menggunakan dua struktur percabangan IF-THEN-ELSE bertingkat untuk menentukan fase zat. Percabangan pertama menentukan apakah suhu lebih besar dari atau sama dengan 100°C (titik didih air). Jika ya, fase zat adalah "Gas". Jika tidak, percabangan kedua menentukan apakah tekanan lebih kecil dari atau sama dengan 101.3 kPa (tekanan atmosfer). Jika ya, fase zat adalah "Padat". Jika tidak, fase zat adalah "Cair".

```
program MenghitungEnergiKinetikGasIdeal;

const
  kb = 1.380649e-23; // Konstanta Boltzmann (J/K)

var
  massa: real;
  suhu: real;
  jenisGas: char;
  kecepatanRataRata: real;
  energiKinetik: real;

begin
  Write('Masukkan massa gas (kg): ');
  ReadLn(massa);
  Write('Masukkan suhu gas (°C): ');
  ReadLn(suhu);
  Write('Masukkan jenis gas (H2, He, N2, O2): ');
  ReadLn(jenisGas);

  case jenisGas of
    'H2':
      kecepatanRataRata := sqrt((8 * kb * (suhu + 273.15)) / massa);
```

```
    'He':
      kecepatanRataRata := sqrt((3 * kb * (suhu + 273.15)) / massa);
    'N2':
      kecepatanRataRata := sqrt((5 * kb * (suhu + 273.15)) / massa);
    'O2':
      kecepatanRataRata := sqrt((5 * kb * (suhu + 273.15)) / massa);
    else:
      WriteLn('Jenis gas tidak valid.');
      Stop;
  end;

  energiKinetik := 0.5 * massa * kecepatanRataRata * kecepatanRataRata;

  WriteLn('Energi kinetik gas ideal (J): ', energiKinetik);
end.
```

2. Menghitung Kapasitas Panas Gas Ideal

Penjelasan:

Kapasitas panas gas ideal dapat dihitung dengan rumus yang berbeda tergantung pada jenis proses:

* Proses isobarik (tekanan konstan): `Cv = (3/2) * R`
* Proses isokhorik (volume konstan): `Cv = (5/2) * R`

Program Pascal:

```
program MenghitungKapasitasPanasGasIdeal;

const
  r = 8.314; // Konstanta gas ideal (J/(mol·K))

var
  n: real;
  jenisProses: char;
  kapasitasPanas: real;

begin
```

```
  Write('Masukkan jumlah mol gas (mol): ');
  ReadLn(n);
  Write('Masukkan jenis proses (isobarik, isokhorik): ');
  ReadLn(jenisProses);

  case jenisProses of
    'isobarik':
      kapasitasPanas := (3/2) * r * n;
    'isokhorik':
      kapasitasPanas := (5/2) * r * n;
    else:
      WriteLn('Jenis proses tidak valid.');
      Stop;
  end;

  WriteLn('Kapasitas panas gas ideal (J/(mol·K)): ', kapasitasPanas);
end.
```

Penjelasan:

Program ini menggunakan percabangan `CASE` untuk menghitung kapasitas panas gas ideal berdasarkan jenis proses. Setiap klausa `CASE` berisi rumus yang sesuai untuk jenis proses yang berbeda.Klausa `ELSE` digunakan untuk menangani jenis proses yang tidak valid.

3. Menghitung Efisiensi Mesin Siklus Otto

Penjelasan:

Efisiensi mesin siklus Otto dapat dihitung dengan rumus `η = 1 - (rc - 1) / rc`, di mana:

- `η` adalah efisiensi termodinamika (%)
- `rc` adalah rasio kompresi (volume awal / volume akhir)

Program Pascal:

```
program MenghitungEfisiensiOtto;

var
  rasioKompresi: real;
  efisiensi: real;
```

```
begin
  Write('Masukkan rasio kompresi: ');
  ReadLn(rasioKompresi);

  if rasioKompresi <= 1 then
    WriteLn('Rasio kompresi tidak valid.')
  else
    efisiensi := 1 - (rasioKompresi - 1) / rasioKompresi;
    WriteLn('Efisiensi mesin siklus Otto: ', efisiensi * 100, '%');
  end;
end.
```

Penjelasan:

Program ini menggunakan percabangan `IF-THEN-ELSE` untuk memastikan bahwa rasio kompresi valid sebelum menghitung efisiensi. Jika rasio kompresi valid, program menghitung efisiensi dengan rumus yang sesuai. Jika rasio kompresi tidak valid, program menampilkan pesan peringatan.

4. Menghitung Entropi Reaksi Kimia

Penjelasan:

Entropi reaksi kimia dapat dihitung dengan rumus `ΔS = ΣS(produk) - ΣS(reaktan)`, di mana:

- `ΔS` adalah entropi reaksi (J/K·mol)
- `S(produk)` adalah entropi molar produk (J/K·mol)
- `S(reaktan)` adalah entropi molar reaktan (J/K·mol)

Program Pascal:

```
program MenghitungEntropiReaksi;

const
  molAir = 18.01528; // Massa molar air (g/mol)

var
  jumlahMolReaktan: real;
  jumlahMolProduk: real;
```

```
  entropiReaksi: real;

begin
  Write('Masukkan jumlah mol reaktan: ');
  ReadLn(jumlahMolReaktan);
  Write('Masukkan jumlah mol produk: ');
  ReadLn(jumlahMolProduk);

  entropiReaksi := (jumlahMolProduk * (-285.834) - jumlahMolReaktan *
(70.918)) / molAir;
  WriteLn('Entropi reaksi (J/K·mol): ', entropiReaksi);
end.
```

C. Rangkuman

Pada bab ini, Anda telah mempelajari struktur percabangan dalam pemrograman Pascal. Struktur percabangan digunakan untuk membuat keputusan dalam program berdasarkan kondisi tertentu. Anda telah belajar cara menggunakan struktur percabangan `IF-THEN-ELSE` dan `CASE` untuk membuat keputusan sederhana dan kompleks dalam program.

D. Test Formatif

1. Jelaskan apa yang dimaksud dengan struktur percabangan.
2. Sebutkan dua jenis struktur percabangan dalam Pascal.
3. Jelaskan fungsi struktur percabangan `IF-THEN-ELSE`.
5. Berikan contoh penggunaan struktur percabangan `CASE` dalam Pascal.
6. Tuliskan program Pascal yang menggunakan struktur percabangan `IF-THEN-ELSE` untuk menghitung nilai mutlak dari sebuah bilangan.

E. Umpan Balik

Setelah menyelesaikan bab ini, diharapkan Anda dapat memahami dan menerapkan struktur percabangan dalam pemrograman Pascal. Jika Anda masih memiliki pertanyaan atau kesulitan, jangan ragu untuk bertanya kepada dosen atau pengajar.

F. Tindak Lanjut

Pelajari lebih lanjut tentang struktur percabangan yang lebih kompleks.Cobalah untuk menulis program Pascal yang lebih kompleks

yang menggunakan struktur percabangan. Ikuti pelatihan atau kursus pemrograman Pascal untuk meningkatkan kemampuan Anda.

Kunci Jawaban Formatif:

1. Struktur percabangan adalah struktur kontrol yang memungkinkan program untuk membuat keputusan dan menjalankan kode yang berbeda berdasarkan kondisi tertentu.
2. Dua jenis struktur percabangan dalam Pascal adalah `IF-THEN-ELSE` dan `CASE`.
3. Struktur percabangan `IF-THEN-ELSE` digunakan untuk membuat keputusan sederhana dengan dua kemungkinan hasil.
4. Contoh penggunaan struktur percabangan `CASE` dalam Pascal adalah untuk membedakan pilihan menu berdasarkan karakter yang diinputkan oleh pengguna.
5. Berikut adalah program Pascal yang menggunakan struktur percabangan `IF-THEN-ELSE` untuk menghitung nilai mutlak dari sebuah bilangan:

```
program MenghitungNilaiMutlak;

var
  bilangan: integer;
  nilaiMutlak: integer;

begin
  Write('Masukkan bilangan: ');
  ReadLn(bilangan);

  if bilangan >= 0 then
    nilaiMutlak := bilangan
  else
    nilaiMutlak := -bilangan;

  WriteLn('Nilai mutlak dari ', bilangan, ' adalah: ', nilaiMutlak);
end.
```

BAB V.
PERULANGAN

A. Struktur Perulangan

Dalam bahasa Pascal, terdapat tiga struktur perulangan yang dapat digunakan, yaitu:

a. erulangan `while`
b. Perulangan `for`
c. Perulangan `do-while`

Perulangan `while`

Perulangan `while` digunakan untuk mengeksekusi blok kode berulang kali selama kondisi yang diberikan bernilai `true`.

```
while kondisi do
begin
  blok kode
end;
```

Contoh pascal

```
program MenghitungBilanganGanjil;

var
  i: integer;

begin
  i := 1;

  while i <= 10 do
  begin
    WriteLn(i, ' ');
    i := i + 2;
  end;
end.
```

Program di atas akan mencetak bilangan ganjil dari 1 hingga 10.

Perulangan `for`

Perulangan `for` digunakan untuk mengeksekusi blok kode berulang kali dengan jumlah yang telah ditentukan.

```
for variabel := nilaiAwal step nilaiLangkah sampai nilaiAkhir do
begin
  blok kode
end;
```

Contoh pascal :

```
program MenjumlahkanBilangan;

var
  i: integer;
  sum: integer;

begin
  sum := 0;

  for i := 1 to 10 do
  begin
    sum := sum + i;
  end;

  WriteLn('Jumlah bilangan dari 1 hingga 10 adalah: ', sum);
end.
```

Program di atas akan menjumlahkan bilangan dari 1 hingga 10 dan menampilkan hasilnya.

Perulangan `do-while`

Perulangan `do-while` mirip dengan perulangan `while`, tetapi blok kode dieksekusi terlebih dahulu sebelum kondisi diperiksa.

```
do
begin
  blok kode
end
while kondisi;
```

Contoh pascal

```
program MenebakAngka;

var
  tebakan: integer;
  angkaRahasia: integer;

begin
  angkaRahasia := Random(1, 10);

  Write('Tebaklah angka antara 1 dan 10: ');
  ReadLn(tebakan);

  do
  begin
    if tebakan = angkaRahasia then
    begin
      WriteLn('Tebakan Anda benar!');
      Break;
    end
    else
    begin
      WriteLn('Tebakan Anda salah. Coba lagi: ');
      ReadLn(tebakan);
    end;
  end;
end.
```

Program di atas akan meminta pengguna untuk menebak angka antara 1 dan 10. Program akan terus berulang sampai pengguna menebak angka yang benar.

B. Aplikasi Perulangan

Perulangan dapat digunakan untuk menyelesaikan berbagai masalah, seperti:

a. Menghitung nilai faktorial suatu bilangan.
b. Menampilkan bilangan prima dari 1 hingga n.
c. Menjumlahkan nilai elemen dalam sebuah array.
d. Mencari nilai maksimum atau minimum dalam sebuah array.

Berikut adalah beberapa contoh aplikasi perulangan dalam bahasa Pascal:

Contoh 1: Menghitung Nilai Faktorial Suatu Bilangan

```
program MenghitungFaktorial;

var
  n: integer;
  i: integer;
  faktorial: integer;

begin
  Write('Masukkan nilai n: ');
  ReadLn(n);

  if n < 0 then
    WriteLn('Nilai n tidak boleh negatif.')
  else
  begin
    faktorial := 1;

    for i := 1 to n do
      faktorial := factorial * i;

    WriteLn('Faktorial dari ', n, ' adalah: ', faktorial);
  end;
end.
```

Contoh 2: Menampilkan Bilangan Prima dari 1 hingga n

```
program MenampilkanBilanganPrima;

var
  n: integer;
```

```
  i: integer;
  prima: boolean;

begin
  Write('Masukkan nilai n: ');
  ReadLn(n);

  if n < 2 then
    WriteLn('Tidak ada bilangan prima di bawah 2.')
  else
  begin
    WriteLn('Bilangan prima dari 1 hingga ', n, ': ');

    for i := 2 to n do
    begin
      prima := true;

      for j := 2 to i - 1 do
      begin
        if i mod j = 0 then
        begin
          prima := false;
          Break;
        end;
      end;

      if prima then
        Write(i, ' ');
    end;
  end;
end.
```

Contoh 3: Menjumlahkan Nilai Elemen dalam Sebuah Array

```
program MenjumlahkanElemenArray;
```

```
var
  i: integer;
  sum: integer;
  array: array[1..10] of integer;

begin
  for i := 1 to 10 do
  begin
    Write('Masukkan nilai elemen ', i, ': ');
    ReadLn(array[i]);
  end;

  sum := 0;

  for i := 1 to 10 do
    sum := sum + array[i];

  WriteLn('Jumlah nilai elemen dalam array adalah: ', sum);
end.
```

Contoh 4: Mencari Nilai Maksimum dalam Sebuah Array

```
program MencariNilaiMaksimum;

var
  i: integer;
  max: integer;
  array: array[1..10] of integer;

begin
  for i := 1 to 10 do
  begin
    Write('Masukkan nilai elemen ', i, ': ');
    ReadLn(array[i]);
  end;
```

```
  max := array[1];

  for i := 2 to 10 do
    if array[i] > max then
      max := array[i];

  WriteLn('Nilai maksimum dalam array adalah: ', max);
end.
```

Contoh 5: Mencetak Bintang dalam Pola Segitiga

```
program MencetakBintangSegitiga;

var
  i: integer;
  j: integer;

begin
  for i := 1 to 5 do
  begin
    for j := 1 to i do
      Write('*');
    WriteLn;
  end;
end.
```

C. Rangkuman

Perulangan merupakan konsep penting dalam pemrograman yang memungkinkan kita untuk mengeksekusi blok kode berulang kali. Dalam bahasa Pascal, terdapat tiga struktur perulangan yang dapat digunakan, yaitu: perulangan `while`, perulangan `for`, dan perulangan `do-while`. Perulangan dapat digunakan untuk menyelesaikan berbagai masalah, seperti menghitung nilai faktorial suatu bilangan, menampilkan bilangan prima, menjumlahkan nilai elemen dalam sebuah array, dan mencari nilai maksimum atau minimum dalam sebuah array.

D. Test Formatif

1. Jelaskan apa yang dimaksud dengan perulangan dalam bahasa Pascal.
2. Sebutkan tiga struktur perulangan yang terdapat dalam bahasa Pascal.

2. Berikan contoh penggunaan perulangan `while` untuk menghitung nilai faktorial suatu bilangan.
3. Berikan contoh penggunaan perulangan `for` untuk menampilkan bilangan prima dari 1 hingga n.
4. Berikan contoh penggunaan perulangan `do-while` untuk mencari nilai maksimum dalam sebuah array.

E. Umpan Balik

a. Apakah materi bab ini sudah jelas dan mudah dipahami?
b. Apakah contoh-contoh yang diberikan sudah cukup membantu?
c. Apakah ada bagian materi yang masih belum dipahami?

F. Tindak Lanjut

a. Cobalah untuk membuat program Pascal yang menggunakan perulangan untuk menyelesaikan masalah-masalah lain.
b. Carilah informasi lebih lanjut tentang perulangan dalam bahasa Pascal.
c. Diskusikan dengan teman atau dosen tentang penggunaan perulangan dalam pemrograman.

G. Kunci Jawaban Formatif

1. Perulangan dalam bahasa Pascal adalah sebuah struktur program yang memungkinkan eksekusi blok kode berulang kali berdasarkan suatu kondisi. Hal ini berguna untuk melakukan proses yang sama berulang kali tanpa harus menulis kode yang sama berulang kali.
2. Tiga Struktur Perulangan dalam Bahasa Pascal
   a. Perulangan `while`: Digunakan untuk mengulangi blok kode selama kondisi yang ditentukan bernilai `true`.
   b. Perulangan `for`: Digunakan untuk mengulangi blok kode berdasarkan rentang nilai yang ditentukan.
   c. Perulangan `repeat until`: Digunakan untuk mengulangi blok kode, kemudian mengecek kondisi. Jika kondisi bernilai `true`, perulangan akan berhenti.
3. Contoh Penggunaan Perulangan `while` untuk Menghitung Faktorial.

```
program Faktorial;

var
  n: integer;
  i: integer;
  faktorial: integer;
```

```
begin
  Write('Masukkan nilai n: ');
  ReadLn(n);

  i := 1;
  faktorial := 1;

  while i <= n do
  begin
    faktorial := faktorial * i;
    i := i + 1;
  end;

  WriteLn('Faktorial dari ', n, ' adalah: ', faktorial);
end.
```

4. Contoh Penggunaan Perulangan `for` untuk Menampilkan Bilangan Prima

```
program BilanganPrima;
var
  n: integer;
  i: integer;
  prima: boolean;

begin
  Write('Masukkan nilai n: ');
  ReadLn(n);

  WriteLn('Bilangan prima dari 1 hingga ', n, ': ');

  for i := 2 to n do
  begin
    prima := true;

    for j := 2 to i - 1 do
```

```
  begin
    if i mod j = 0 then
    begin
      prima := false;
      Exit;
    end;
  end;

  if prima then
    Write(i, ' ');
 end;
end.
```

Program ini mencari dan menampilkan bilangan prima dari 1 hingga n.

- ❖ Pengguna memasukkan nilai n.
- ❖ Program mencari bilangan prima dari 2 hingga n.
- ❖ Untuk setiap bilangan, dicek apakah habis dibagi bilangan 2 hingga n-1.
- ❖ Jika tidak habis dibagi, maka bilangan tersebut adalah prima dan ditampilkan.
- ❖ Dua perulangan for bersarang digunakan untuk mencapai tujuan ini.

5. Contoh penggunaan perulangan `do-while` untuk mencari nilai maksimum dalam sebuah array.

```
program MencariNilaiMaksimum;

var
 i: integer;
 max: integer;
 array: array[1..10] of integer;

begin
 for i := 1 to 10 do
 begin
  Write('Masukkan nilai elemen ', i, ': ');
  ReadLn(array[i]);
 end;
```

```
  i := 1;
  max := array[1];

  do
  begin
    if array[i] > max then
      max := array[i];
    i := i + 1;
  end
  while i <= 10;

  WriteLn('Nilai maksimum dalam array adalah: ', max);
end.
```

Penjelasan:

Perulangan `while`: Digunakan untuk menghitung faktorial. Di dalam perulangan, variabel `i` diulang dari 1 hingga `n` dan dikalikan dengan variabel `faktorial`. Perulangan `for`: Digunakan untuk menampilkan bilangan prima. Di dalam perulangan, variabel `i` diulang dari 2 hingga `n`. Di dalam perulangan `for` lain, dilakukan pengecekan apakah `i` habis dibagi bilangan lain (2 hingga `i` - 1). Jika tidak habis dibagi, maka `i` adalah bilangan prima dan ditampilkan.

BAB VI.
FUNGSI DAN PROSEDUR

A. Konsep Fungsi dan Prosedur

Fungsi dan prosedur adalah modul program yang memungkinkan programmer untuk memecah program menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola.

Fungsi adalah modul program yang menghasilkan nilai kembalian. Nilai kembalian dapat berupa tipe data apa pun.

Prosedur adalah modul program yang tidak menghasilkan nilai kembalian. Prosedur biasanya digunakan untuk melakukan tugas-tugas tertentu, seperti mencetak output ke layar atau membaca input dari pengguna.

B. Menulis Fungsi dan Prosedur

Fungsi dan prosedur ditulis dengan menggunakan kata kunci `function` atau `procedure` diikuti dengan nama fungsi atau prosedur, parameter (jika ada), dan blok kode.

Contoh program pascal

```
function LuasPersegi(sisi: real): real;
begin
  LuasPersegi := sisi * sisi;
end;

procedure CetakHalo;
begin
  WriteLn('Halo!');
end;
```

C. Parameter dan Nilai Kembalian Fungsi

Fungsi dapat memiliki parameter yang digunakan untuk menerima nilai dari program pemanggil. Nilai kembalian fungsi dapat digunakan untuk mengembalikan nilai ke program pemanggil. Contoh program pascal sebagai berikut :

```
function LuasPersegi(sisi: real): real;
var
  luas: real;
begin
```

```
  luas := sisi * sisi;
  LuasPersegi := luas;
end;
```

Pada contoh di atas, fungsi `LuasPersegi` memiliki parameter `sisi` yang digunakan untuk menerima nilai sisi persegi. Nilai luas persegi dihitung dan disimpan dalam variabel `luas`. Nilai `luas` kemudian dikembalikan sebagai nilai kembalian fungsi.

D. Penggunaan Fungsi dan Prosedur Pustaka Standar Pascal

Bahasa Pascal menyediakan beberapa fungsi dan prosedur pustaka standar yang dapat digunakan oleh programmer. Fungsi dan prosedur pustaka standar ini dapat digunakan untuk berbagai tugas, seperti input/output, manipulasi string, dan operasi matematika.Contoh program pascal sebagai berikut :

```
program MenjumlahkanBilangan;

var
  a: integer;
  b: integer;
  sum: integer;

begin
  Write('Masukkan nilai a: ');
  ReadLn(a);
  Write('Masukkan nilai b: ');
  ReadLn(b);

  sum := a + b;

  WriteLn('Jumlah ', a, ' dan ', b, ' adalah: ', sum);
end.
```

Pada contoh di atas, program menggunakan fungsi `ReadLn` untuk membaca input dari pengguna dan fungsi `WriteLn` untuk mencetak output ke layar.

Berikut adalah contoh-contoh program Pascal untuk perhitungan fisika berkaitan dengan listrik dan magnet (Bab 6) yang dikaitkan dengan fungsi dan prosedur:

1. Menghitung Gaya Lorentz pada Muatan Bergerak dalam Medan Magnet (dengan Fungsi)

```
program GayaLorentz;

var
  q: real;   { Muatan (Coulomb) }
  v: real;   { Kecepatan (meter per sekon) }
  B: real;   { Kuat medan magnet (Tesla) }
  theta: real; { Sudut antara vektor kecepatan dan medan magnet }

function HitungGayaLorentz(q: real; v: real; B: real; theta: real): real;
var
  Fx: real;  { Gaya Lorentz dalam arah x (Newton) }
  Fy: real;  { Gaya Lorentz dalam arah y (Newton) }
begin
  Fx := q * v * B * Sin(theta);   { Hitung gaya Lorentz dalam arah x }
  Fy := q * v * B * Cos(theta);   { Hitung gaya Lorentz dalam arah y }

  HitungGayaLorentz := Sqrt(Fx^2 + Fy^2);    { Hitung besarnya gaya
Lorentz }
end;

begin
  Write('Masukkan nilai muatan (Coulomb): ');
  ReadLn(q);
  Write('Masukkan nilai kecepatan (meter per sekon): ');
  ReadLn(v);
  Write('Masukkan nilai kuat medan magnet (Tesla): ');
  ReadLn(B);
  Write('Masukkan nilai sudut (derajat): ');
  ReadLn(theta);

  theta := theta * pi / 180;   { Konversi sudut dari derajat ke radian }
```

```
  WriteLn('Gaya Lorentz: ', HitungGayaLorentz(q, v, B, theta):4:2);
end.
```

Penjelasan:

Pada contoh ini, fungsi `HitungGayaLorentz` digunakan untuk menghitung gaya Lorentz pada muatan bergerak dalam medan magnet. Fungsi ini menerima empat parameter: muatan (q), kecepatan (v), kuat medan magnet (B), dan sudut (theta). Fungsi ini menghitung gaya Lorentz dalam arah x dan y, dan kemudian menghitung besarnya gaya Lorentz dengan menggunakan rumus Pythagoras.

2. Menghitung Induktansi Solenoid (dengan Prosedur)

```
program InduktansiSolenoid;

var
  N: integer; { Jumlah lilitan }
  l: real;   { Panjang solenoid (meter) }
  r: real;   { Jari-jari solenoid (meter) }
  A: real;   { Luas penampang solenoid (meter persegi) }

procedure HitungInduktansi(N: integer; l: real; r: real; A: real; var L:
real);
begin
  L := (mu_0 * N^2 * A) / (2 * l);   { Hitung induktansi solenoid }
end;

begin
  Write('Masukkan nilai jumlah lilitan: ');
  ReadLn(N);
  Write('Masukkan nilai panjang solenoid (meter): ');
  ReadLn(l);
  Write('Masukkan nilai jari-jari solenoid (meter): ');
  ReadLn(r);
  Write('Masukkan nilai luas penampang solenoid (meter persegi): ');
  ReadLn(A);
```

```
  var L: real;   { Induktansi solenoid (Henry) }

  HitungInduktansi(N, l, r, A, L);

  WriteLn('Induktansi solenoid: ', L:4:2);
end.
```

Penjelasan:

Pada contoh ini, prosedur `HitungInduktansi` digunakan untuk menghitung induktansi solenoid. Prosedur ini menerima lima parameter: jumlah lilitan (N), panjang solenoid (l), jari-jari solenoid (r), luas penampang solenoid (A), dan variabel untuk menyimpan nilai induktansi solenoid (L). Prosedur ini menghitung induktansi solenoid dengan menggunakan rumus yang telah dijelaskan sebelumnya.

3. Menghitung Gaya Listrik Antara Dua Muatan Titik (dengan Fungsi)

```
program GayaListrik;

var
  q1: real;  { Muatan pertama (Coulomb) }
  q2: real;  { Muatan kedua (Coulomb) }
  r: real;   { Jarak antara dua muatan (meter) }

function HitungGayaListrik(q1: real; q2: real; r: real): real;
begin
  F := 9e9 * q1 * q2 / (r^2);   { Hitung gaya listrik }
end;

begin
  Write('Masukkan nilai muatan pertama (Coulomb): ');
  ReadLn(q1);
  Write('Masukkan nilai muatan kedua (Coulomb): ');
  ReadLn(q2);
  Write('Masukkan nilai jarak antara dua muatan (meter): ');
  ReadLn(r);
```

```
  WriteLn('Gaya listrik: ', HitungGayaListrik(q1, q2, r):4:2);
end.
```

Penjelasan:

Pada contoh ini, fungsi `HitungGayaListrik` digunakan untuk menghitung gaya listrik antara dua muatan titik. Fungsi ini menerima tiga parameter: muatan pertama (q1), muatan kedua (q2), dan jarak antara dua muatan (r). Fungsi ini menghitung gaya listrik dengan menggunakan rumus Coulomb.

4. Menghitung Tegangan Listrik pada Resistor (dengan Prosedur)

```
program TeganganListrik;

var
  I: real;   { Arus (ampere) }
  R: real;   { Resistensi (ohm) }

procedure HitungTegangan(I: real; R: real; var V: real);
begin
  V := I * R;   { Hitung tegangan listrik }
end;

begin
  Write('Masukkan nilai arus (ampere): ');
  ReadLn(I);
  Write('Masukkan nilai resistensi (ohm): ');
  ReadLn(R);

  var V: real;   { Tegangan listrik (volt) }

  HitungTegangan(I, R, V);

  WriteLn('Tegangan listrik: ', V:4:2);
end.
```

Penjelasan:

Pada contoh ini, prosedur `HitungTegangan` digunakan untuk menghitung tegangan listrik pada resistor. Prosedur ini menerima tiga parameter: arus (I), resistensi (R), dan variabel untuk menyimpan nilai tegangan listrik (V). Prosedur ini menghitung tegangan listrik dengan menggunakan hukum Ohm.

5. Menghitung Daya Listrik (dengan Fungsi)

```
program DayaListrik;

var
  V: real;   { Tegangan listrik (volt) }
  I: real;   { Arus (ampere) }

function HitungDaya(V: real; I: real): real;
begin
  P := V * I;   { Hitung daya listrik }
end;

begin
  Write('Masukkan nilai tegangan listrik (volt): ');
  ReadLn(V);
  Write('Masukkan nilai arus (ampere): ');
  ReadLn(I);

  WriteLn('Daya listrik: ', HitungDaya(V, I):4:2);
end.
```

Penjelasan:

Pada contoh ini, fungsi `HitungDaya` digunakan untuk menghitung daya listrik. Fungsi ini menerima dua parameter: tegangan listrik (V) dan arus (I). Fungsi ini menghitung daya listrik dengan menggunakan rumus P = V * I.

E. Rangkuman

Fungsi dan prosedur merupakan modul program yang memungkinkan programmer untuk memecah program menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola. Hal ini dapat

meningkatkan keterbacaan, modularitas, dan reusabilitas kode. Fungsi menghasilkan nilai kembalian, sedangkan prosedur tidak. Fungsi dan prosedur dapat memiliki parameter yang digunakan untuk menerima nilai dari program pemanggil. Bahasa Pascal menyediakan beberapa fungsi dan prosedur pustaka standar yang dapat digunakan oleh programmer.

F. Test Formatif

1. Jelaskan perbedaan antara fungsi dan prosedur.
2. Berikan contoh penggunaan fungsi untuk menghitung luas persegi panjang.
3. Berikan contoh penggunaan prosedur untuk mencetak pesan "Halo!" ke layar.
4. Jelaskan bagaimana cara menggunakan fungsi dan prosedur pustaka standar Pascal.

G. Umpan Balik

a. Apakah materi bab ini sudah jelas dan mudah dipahami?
b. Apakah contoh-contoh yang diberikan sudah cukup membantu?
c. Apakah ada bagian materi yang masih belum dipahami?

H. Tindak Lanjut

a. Cobalah untuk membuat program Pascal yang menggunakan fungsi dan prosedur untuk menyelesaikan masalah-masalah lain.
b. Carilah informasi lebih lanjut tentang fungsi dan prosedur dalam bahasa Pascal.
c. Diskusikan dengan teman atau dosen tentang penggunaan fungsi dan prosedur dalam pemrograman.

I. Kunci Jawaban Formatif

1. Perbedaan utama antara fungsi dan prosedur adalah:
   - ❖ Nilai kembalian: Fungsi menghasilkan nilai kembalian, sedangkan prosedur tidak.
   - ❖ Penggunaan: Fungsi biasanya digunakan untuk menghitung nilai atau melakukan operasi tertentu, sedangkan prosedur biasanya digunakan untuk melakukan tugas-tugas tertentu, seperti mencetak output ke layar atau membaca input dari pengguna.
2. Berikan contoh penggunaan fungsi untuk menghitung luas persegi panjang.

Jawaban:

```
function LuasPersegiPanjang(panjang: real; lebar: real): real;
begin
  LuasPersegiPanjang := panjang * lebar;
end;
```

3. Berikan contoh penggunaan prosedur untuk mencetak pesan "Halo!" ke layar.

Jawaban:

```
procedure CetakHalo;
begin
  WriteLn('Halo!');
end;
```

4. Jelaskan bagaimana cara menggunakan fungsi dan prosedur pustaka standar Pascal.

Jawaban:

Fungsi dan prosedur pustaka standar Pascal dapat digunakan dengan cara memanggil namanya diikuti dengan parameternya (jika ada). Contohnya:

```
program MenjumlahkanBilangan;

var
  a: integer;
  b: integer;
  sum: integer;

begin
  Write('Masukkan nilai a: ');
  ReadLn(a);
  Write('Masukkan nilai b: ');
  ReadLn(b);

  sum := a + b;

  WriteLn('Jumlah ', a, ' dan ', b, ' adalah: ', sum);
end.
```

Pada contoh di atas, program menggunakan fungsi `ReadLn` untuk membaca input dari pengguna dan fungsi `WriteLn` untuk mencetak output ke layar.

BAB VII.
ARRAY

A. Konsep Array

Array adalah struktur data yang memungkinkan programmer untuk menyimpan koleksi nilai-nilai dengan tipe data yang sama. Array didefinisikan dengan menggunakan nama array, tipe data elemen array, dan dimensi array.Contoh program pascal

```
var
  nilai: array[1..10] of integer;
```

Pada contoh di atas, array `nilai` didefinisikan untuk menyimpan 10 nilai integer. Indeks array `nilai` dimulai dari 1 hingga 10.

B. Mengakses Elemen Array

Elemen-elemen array diakses dengan menggunakan indeks. Indeks array adalah bilangan bulat yang menunjukkan posisi elemen dalam array. Contoh program pascal

```
nilai[1] := 10;
nilai[5] := 20;
```

Pada contoh di atas, nilai 10 disimpan pada elemen array `nilai` dengan indeks 1, dan nilai 20 disimpan pada elemen array `nilai` dengan indeks 5.

C. Operasi-operasi Dasar pada Array

Ada beberapa operasi dasar yang dapat dilakukan pada array, seperti:

a. Mencari: Mencari nilai tertentu dalam array.
b. Menyusun: Mengurutkan elemen-elemen array berdasarkan nilai-nilainya.
c. Menghapus: Menghapus elemen tertentu dari array.

D. Array Dua Dimensi

Array dua dimensi adalah array yang memiliki dua dimensi. Dimensi pertama adalah baris, dan dimensi kedua adalah kolom. Contoh program pascal

```
var
  matriks: array[1..10, 1..5] of integer;
```

Pada contoh di atas, array `matriks` didefinisikan untuk menyimpan 10 baris dan 5 kolom nilai integer. Berikut ini akan disajikan 5 Contoh Perhitungan Pascal dengan Array untuk Fluida.

1. Menghitung Tekanan Fluida Statis dengan Array (Hukum Pascal)

```
program TekananPascal;

var
  n: integer;   { Jumlah titik pengukuran }
  tekanan: array[1..n] of real;   { Tekanan pada setiap titik pengukuran }
  tekanan_awal: real;   { Tekanan awal }

begin
  Write('Masukkan jumlah titik pengukuran: ');
  ReadLn(n);

  Write('Masukkan tekanan awal: ');
  ReadLn(tekanan_awal);

  for i := 1 to n do
  begin
    Write('Masukkan tekanan pada titik ', i, ': ');
    ReadLn(tekanan[i]);
  end;

  for i := 1 to n do
  begin
    if tekanan[i] <> tekanan_awal then
    begin
      WriteLn('Tekanan pada titik ', i, ' tidak sama dengan tekanan awal.');
      break;
    end;
  end;

  WriteLn('Tekanan pada semua titik sama dengan tekanan awal.');
end.
```

Penjelasan:

Program ini menghitung tekanan fluida statis dengan menggunakan Hukum Pascal. Hukum Pascal menyatakan bahwa tekanan yang diberikan pada fluida tertutup akan diteruskan ke segala arah tanpa mengalami perubahan. Program ini menerima input jumlah titik pengukuran dan tekanan awal, kemudian meminta pengguna untuk memasukkan tekanan pada setiap titik pengukuran. Program ini kemudian memeriksa apakah tekanan pada semua titik pengukuran sama dengan tekanan awal. Jika ya, maka Hukum Pascal berlaku.

2. Menghitung Debit Fluida dengan Array (Persamaan Kontinuitas)

```
program DebitFluida;

var
  pi: real;   { Nilai pi (3.14159) }
  jariJari: array[1..n] of real;  { Jari-jari pipa pada setiap titik pengukuran
}
  kecepatan: array[1..n] of real;   { Kecepatan aliran fluida pada setiap
titik pengukuran }
  debit: array[1..n] of real;  { Debit fluida pada setiap titik pengukuran }

begin
  pi := 3.14159;
  Write('Masukkan jumlah titik pengukuran: ');
  ReadLn(n);

  for i := 1 to n do
  begin
    Write('Masukkan jari-jari pipa pada titik ', i, ': ');
    ReadLn(jariJari[i]);
    Write('Masukkan kecepatan aliran fluida pada titik ', i, ': ');
    ReadLn(kecepatan[i]);
  end;

  for i := 1 to n do
  begin
    debit[i] := pi * jariJari[i] ^ 2 * kecepatan[i];
```

```
    WriteLn('Debit fluida pada titik ', i, ': ', debit[i]:4:2);
  end;
end.
```

Penjelasan:

Program ini menghitung debit fluida dengan menggunakan Persamaan Kontinuitas. Persamaan Kontinuitas menyatakan bahwa debit fluida yang masuk ke suatu sistem harus sama dengan debit fluida yang keluar dari sistem tersebut. Program ini menerima input jumlah titik pengukuran, jari-jari pipa, dan kecepatan aliran fluida pada setiap titik pengukuran. Program ini kemudian menghitung debit fluida pada setiap titik pengukuran dengan menggunakan rumus debit fluida.

3. Menghitung Gaya Angkat pada Benda Terendam dengan Array (Hukum Archimedes)

```
program GayaAngkat;

var
  n: integer;   { Jumlah benda terendam }
  massa: array[1..n] of real;   { Massa benda terendam }
  volume: array[1..n] of real;   { Volume benda terendam }
  wajahAir: real;   { Ketinggian permukaan air }
  gayaAngkat: array[1..n] of real;    { Gaya angkat pada setiap benda
terendam }

begin
  Write('Masukkan jumlah benda terendam: ');
  ReadLn(n);

  Write('Masukkan ketinggian permukaan air: ');
  ReadLn(wajahAir);

  for i := 1 to n do
  begin
    Write('Masukkan massa benda terendam ', i, ': ');
    ReadLn(massa[i]);
    Write('Masukkan volume benda terendam ', i, ': ');
    ReadLn(volume[i]);
```

```
  end;

  for i := 1 to n do
  begin
    gayaAngkat[i] := massa[i] * g * (wajahAir - volume[i] / (pi * (jariJari[i]
^ 2)));

  WriteLn('Gaya angkat pada benda terendam ', i, ': ',
gayaAngkat[i]:4:2);
  end;
end.
```

Penjelasan:

Program ini menghitung gaya angkat pada benda terendam dengan menggunakan Hukum Archimedes. Hukum Archimedes menyatakan bahwa gaya angkat pada benda terendam sama dengan berat fluida yang didesak oleh benda tersebut. Program ini menerima input jumlah benda terendam, ketinggian permukaan air, massa, dan volume benda terendam. Program ini kemudian menghitung gaya angkat pada setiap benda terendam dengan menggunakan rumus gaya angkat.

4. Menghitung Energi Potensial Fluida dengan Array (Persamaan Energi Potensial Fluida)

```
program EnergiPotensialFluida;

var
  n: integer;  { Jumlah titik pengukuran }
  tinggi: array[1..n] of real;    { Ketinggian fluida pada setiap titik
pengukuran }
  massaJenis: real;  { Massa jenis fluida }
  g: real;  { Percepatan gravitasi }
  energiPotensial: array[1..n] of real;    { Energi potensial fluida pada
setiap titik pengukuran }

begin
  Write('Masukkan jumlah titik pengukuran: ');
  ReadLn(n);
```

```
  Write('Masukkan massa jenis fluida: ');
  ReadLn(massaJenis);
  Write('Masukkan percepatan gravitasi: ');
  ReadLn(g);

  for i := 1 to n do
  begin
    Write('Masukkan ketinggian fluida pada titik ', i, ': ');
    ReadLn(tinggi[i]);
  end;

  for i := 1 to n do
  begin
    energiPotensial[i] := massaJenis * g * tinggi[i];
    WriteLn('Energi potensial fluida pada titik ', i, ': ',
energiPotensial[i]:4:2);
  end;
end.
```

Penjelasan:

Program ini menghitung energi potensial fluida dengan menggunakan Persamaan Energi Potensial Fluida. Persamaan Energi Potensial Fluida menyatakan bahwa energi potensial fluida sama dengan massa jenis fluida dikali percepatan gravitasi dikali ketinggian fluida. Program ini menerima input jumlah titik pengukuran, massa jenis fluida, percepatan gravitasi, dan ketinggian fluida pada setiap titik pengukuran. Program ini kemudian menghitung energi potensial fluida pada setiap titik pengukuran dengan menggunakan rumus energi potensial fluida.

5. Menghitung Energi Kinetik Fluida dengan Array (Persamaan Energi Kinetik Fluida)

```
program EnergiKinetikFluida;

var
  n: integer;   { Jumlah titik pengukuran }
```

```
  kecepatan: array[1..n] of real;   { Kecepatan aliran fluida pada setiap
titik pengukuran }
  massaJenis: real;   { Massa jenis fluida }
  energiKinetik: array[1..n] of real;   { Energi kinetik fluida pada setiap
titik pengukuran }

begin
  Write('Masukkan jumlah titik pengukuran: ');
  ReadLn(n);

  Write('Masukkan massa jenis fluida: ');
  ReadLn(massaJenis);

  for i := 1 to n do
  begin
    Write('Masukkan kecepatan aliran fluida pada titik ', i, ': ');
    ReadLn(kecepatan[i]);
  end;

  for i := 1 to n do
  begin
    energiKinetik[i] := 0.5 * massaJenis * (kecepatan[i] ^ 2);
    WriteLn('Energi kinetik fluida pada titik ', i, ': ', energiKinetik[i]:4:2);
  end;
end.
```

Penjelasan:

Program ini menghitung energi kinetik fluida dengan menggunakan Persamaan Energi Kinetik Fluida. Persamaan Energi Kinetik Fluida menyatakan bahwa energi kinetik fluida sama dengan setengah massa jenis fluida dikali kuadrat kecepatan aliran fluida. Program ini menerima input jumlah titik pengukuran, massa jenis fluida, dan kecepatan aliran fluida pada setiap titik pengukuran. Program ini kemudian menghitung energi kinetik fluida pada setiap titik pengukuran dengan menggunakan rumus energi kinetik fluida.

E. Rangkuman

Array adalah struktur data yang sangat penting dalam bahasa Pascal. Array memungkinkan programmer untuk menyimpan koleksi nilai-nilai dengan tipe data yang sama dengan cara yang efisien dan terstruktur. Array didefinisikan dengan menggunakan nama array, tipe data elemen array, dan dimensi array. Elemen-elemen array diakses dengan menggunakan indeks. Ada beberapa operasi dasar yang dapat dilakukan pada array, seperti mencari, menyusun, dan menghapus elemen. Array dua dimensi adalah array yang memiliki dua dimensi, yaitu baris dan kolom.

F. Test Formatif

1. Jelaskan apa itu array dalam bahasa Pascal.
2. Bagaimana cara mendefinisikan array dalam bahasa Pascal?
2. Bagaimana cara mengakses elemen-elemen array dalam bahasa Pascal?
5. Sebutkan beberapa operasi dasar yang dapat dilakukan pada array.
6. Jelaskan apa itu array dua dimensi.

G. Umpan Balik

a. Apakah materi bab ini sudah jelas dan mudah dipahami? (Jawaban dari mahasiswa)
b. Apakah contoh-contoh program yang diberikan sudah cukup membantu? (Jawaban dari mahasiswa)
c. Apakah ada bagian materi yang masih belum dipahami? (Jawaban dari mahasiswa)

H. Tindak Lanjut

a. Cobalah untuk membuat program Pascal yang menggunakan array untuk menyelesaikan masalah-masalah lain.
b. Carilah informasi lebih lanjut tentang array dalam bahasa Pascal.
c. Diskusikan dengan teman atau dosen tentang penggunaan array dalam pemrograman.

I. Kunci Jawaban Tes Formatif

1. Jelaskan apa itu array dalam bahasa Pascal.

Array dalam bahasa Pascal adalah struktur data yang memungkinkan programmer untuk menyimpan koleksi nilai-nilai dengan tipe data yang sama. Array didefinisikan dengan menggunakan nama array, tipe data elemen array, dan dimensi array.

2. Bagaimana cara mendefinisikan array dalam bahasa Pascal?

Array didefinisikan dengan menggunakan kata kunci `array` diikuti dengan nama array, kurung siku, tipe data elemen array, dan dimensi array. Contoh:

```
var
  nilai: array[1..10] of integer;
```

Pada contoh di atas, array `nilai` didefinisikan untuk menyimpan 10 nilai integer. Indeks array `nilai` dimulai dari 1 hingga 10.

3. Bagaimana cara mengakses elemen-elemen array dalam bahasa Pascal?

Elemen-elemen array diakses dengan menggunakan indeks. Indeks array adalah bilangan bulat yang menunjukkan posisi elemen dalam array. Contoh:

```
nilai[1] := 10;
nilai[5] := 20;
```

Pada contoh di atas, nilai 10 disimpan pada elemen array `nilai` dengan indeks 1, dan nilai 20 disimpan pada elemen array `nilai` dengan indeks 5.

4. Sebutkan beberapa operasi dasar yang dapat dilakukan pada array.

Beberapa operasi dasar yang dapat dilakukan pada array adalah:

a. Mencari: Mencari nilai tertentu dalam array.
b. Menyusun: Mengurutkan elemen-elemen array berdasarkan nilai-nilainya.
c. Menghapus: Menghapus elemen tertentu dari array.

5. Jelaskan apa itu array dua dimensi.

Array dua dimensi adalah array yang memiliki dua dimensi. Dimensi pertama adalah baris, dan dimensi kedua adalah kolom. Contoh:

```
var
  matriks: array[1..10, 1..5] of integer;
```

Pada contoh di atas, array `matriks` didefinisikan untuk menyimpan 10 baris dan 5 kolom nilai integer.

BAB VIII.
STRING

A. Konsep String

String adalah urutan karakter yang diwakili oleh tipe data `string` dalam bahasa Pascal. String dapat didefinisikan dengan menggunakan tanda kutip tunggal (') atau tanda kutip ganda ("). Contoh program pascal sebagai berikut.

```
var
  nama: string;
  pesan: string;

begin
  nama := 'Budi';
  pesan := "Selamat datang di dunia pemrograman Pascal!";
end.
```

Pada contoh di atas, variabel `nama` didefinisikan sebagai string dengan nilai "Budi", dan variabel `pesan` didefinisikan sebagai string dengan nilai "Selamat datang di dunia pemrograman Pascal!".

B. Mengakses dan Memanipulasi Elemen String

Elemen-elemen string dapat diakses dengan menggunakan indeks. Indeks string dimulai dari 1.Contoh program `pascal

```
var
  huruf: char;

begin
  huruf := nama[1];
  WriteLn(huruf);   { Output: B }
end.
```

Pada contoh di atas, variabel `huruf` menyimpan nilai elemen pertama dari string `nama`, yaitu karakter 'B'. String juga dapat dimanipulasi dengan menggunakan berbagai operasi, seperti:

a. Penyambungan string: Digunakan untuk menggabungkan dua string.
b. Pemotongan string: Digunakan untuk mengambil bagian dari string.
c. Pencarian string: Digunakan untuk mencari substring dalam string.

d. Pergantian karakter: Digunakan untuk mengganti karakter dalam string.

C. Fungsi-fungsi String

Bahasa Pascal menyediakan berbagai fungsi string yang dapat digunakan untuk memanipulasi string. Beberapa fungsi string yang umum digunakan adalah:

a. Length(s): Menghitung panjang string `s`.
b. Copy(s1, s2, i, n): Menyalin `n` karakter dari string `s1` ke string `s2` mulai dari indeks `i`.
c. Pos(s1, s2): Mencari posisi substring `s2` dalam string `s1`.
d. Delete(s, i, n): Menghapus `n` karakter dari string `s` mulai dari indeks `i`.
e. Insert(s, i, s2): Menyisipkan string `s2` ke string `s` pada indeks `i`.

D. Membandingkan String

String dapat dibandingkan dengan menggunakan operator perbandingan, seperti:

a. `=`: Membandingkan apakah dua string sama.
b. `<>`: Membandingkan apakah dua string berbeda.
c. `<`: Membandingkan apakah string pertama lebih kecil dari string kedua.
d. `<=`: Membandingkan apakah string pertama lebih kecil dari atau sama dengan string kedua.
e. `>`: Membandingkan apakah string pertama lebih besar dari string kedua.
f. `>=`: Membandingkan apakah string pertama lebih besar dari atau sama dengan string kedua.

Meskipun string umumnya digunakan untuk manipulasi teks, string juga dapat dimanfaatkan untuk perhitungan fisika tertentu, terutama yang melibatkan representasi data teks dan pemrosesan data eksperimen. Berikut adalah 3 contoh perhitungan fisika dengan string:

a. Konversi Satuan dengan String

Seringkali dalam fisika, kita perlu mengonversi nilai antara satuan yang berbeda. String dapat digunakan untuk menyimpan informasi tentang faktor konversi dan nama satuan.

Contoh:

```
program KonversiSatuan;
```

```
var
  nilai: real;
  satuanAsal, satuanBaru: string;
  faktorKonversi: real;

begin
  Write('Masukkan nilai: ');
  ReadLn(nilai);
  Write('Masukkan satuan asal: ');
  ReadLn(satuanAsal);
  Write('Masukkan satuan baru: ');
  ReadLn(satuanBaru);

  if satuanAsal = 'm' and satuanBaru = 'cm' then
  begin
    faktorKonversi := 100;
    nilaiBaru := nilai * faktorKonversi;
    WriteLn(nilai, satuanAsal, ' sama dengan ', nilaiBaru, satuanBaru);
  end
  else if satuanAsal = 'cm' and satuanBaru = 'm' then
  begin
    faktorKonversi := 0.01;
    nilaiBaru := nilai * faktorKonversi;
    WriteLn(nilai, satuanAsal, ' sama dengan ', nilaiBaru, satuanBaru);
  end
  else
  begin
    WriteLn('Konversi satuan tidak tersedia.');
  end;
end.
```

Program ini memungkinkan pengguna untuk memasukkan nilai dan satuannya, kemudian program akan mengonversi nilainya ke satuan baru berdasarkan faktor konversi yang disimpan dalam variabel string.

b. Analisis Data Eksperimen dengan String

Data eksperimen dalam fisika sering kali direpresentasikan dalam bentuk teks, seperti tabel atau daftar. String dapat digunakan untuk menyimpan dan memproses data ini untuk analisis lebih lanjut.

Contoh:

```
program AnalisisData;

var
  data: array[1..10] of string;
  rataRata: real;
  jumlahData: integer;

begin
  jumlahData := 0;
  rataRata := 0;

  for i := 1 to 10 do
  begin
    Write('Masukkan data ', i, ': ');
    ReadLn(data[i]);

    if IsNumeric(data[i]) then
    begin
      jumlahData := jumlahData + 1;
      rataRata := rataRata + StrToVal(data[i]);
    end;
  end;

  if jumlahData > 0 then
  begin
    rataRata := rataRata / jumlahData;
    WriteLn('Rata-rata data: ', rataRata);
  end
  else
  begin
    WriteLn('Data tidak valid.');
```

```
  end;
end.
```

Program ini memungkinkan pengguna untuk memasukkan 10 data eksperimen dalam bentuk string. Program kemudian memeriksa apakah data numerik dan menghitung rata-rata dari data yang valid.

c. Simulasi Gerak dengan String

String dapat digunakan untuk menyimpan informasi tentang posisi, kecepatan, dan percepatan objek yang bergerak dalam simulasi fisika.

Contoh:

```
program SimulasiGerak;

var
  posisi: string;
  kecepatan: string;
  percepatan: string;
  waktu: real;

begin
  Write('Masukkan posisi awal: ');
  ReadLn(posisi);
  Write('Masukkan kecepatan awal: ');
  ReadLn(kecepatan);
  Write('Masukkan percepatan: ');
  ReadLn(percepatan);
  Write('Masukkan waktu: ');
  ReadLn(waktu);

  // Melakukan perhitungan posisi akhir dengan rumus fisika gerak
  // ...

  WriteLn('Posisi akhir: ', posisiAkhir);
end.
```

Program ini memungkinkan pengguna untuk memasukkan informasi awal tentang posisi, kecepatan, dan percepatan objek, serta waktu simulasi. Program kemudian melakukan perhitungan posisi akhir objek dengan menggunakan rumus fisika gerak dan menyimpan hasilnya dalam string.

E. Rangkuman

String adalah urutan karakter yang digunakan untuk mewakili teks atau informasi tekstual. String adalah salah satu tipe data yang paling penting dalam pemrograman, karena banyak aplikasi yang membutuhkan manipulasi teks. Bab ini membahas tentang konsep string dalam bahasa Pascal, termasuk definisi string, akses dan manipulasi elemen string, fungsi-fungsi string, dan perbandingan string.

F. Test Formatif

1. Jelaskan apa itu string dalam bahasa Pascal.
2. Bagaimana cara mendefinisikan string dalam bahasa Pascal?
3. Bagaimana cara mengakses elemen pertama dari string "Belajar Pascal"?
4. Jelaskan apa itu fungsi `Length(s)` dalam bahasa Pascal.
5. Berikan contoh program Pascal yang menggunakan operator `<>` untuk membandingkan dua string.

G. Umpan Balik

a. Apakah materi bab ini sudah jelas dan mudah dipahami?
b. Apakah contoh-contoh program yang diberikan sudah cukup membantu?
c. Apakah ada bagian materi yang masih belum dipahami?

H. Tindak Lanjut

a. Cobalah untuk membuat program Pascal yang menggunakan string untuk menyelesaikan masalah-masalah lain.
b. Carilah informasi lebih lanjut tentang string dalam bahasa Pascal.
c. Diskusikan dengan teman atau dosen tentang penggunaan string dalam pemrograman.

I. Kunci Jawaban Formatif

1. String adalah urutan karakter yang diwakili oleh tipe data `string` dalam bahasa Pascal.
2. String dapat didefinisikan dengan menggunakan tanda kutip tunggal (') atau tanda kutip ganda (").

3. Elemen pertama dari string "Belajar Pascal" dapat diakses dengan menggunakan indeks 1, yaitu karakter 'B'.
4. Fungsi `Length(s)` menghitung panjang string `s`.
5. Berikut adalah contoh program Pascal yang menggunakan operator `<>` untuk membandingkan dua string:

```
program BandingString;
var
  nama1, nama2: string;
begin
  Write('Masukkan nama 1: ');
  ReadLn(nama1);
  Write('Masukkan nama 2: ');
  ReadLn(nama2);

  if nama1 <> nama2 then
  begin
    WriteLn(nama1, ' dan ', nama2, ' berbeda.');
  end
  else
  begin
    WriteLn(nama1, ' dan ', nama2, ' sama.');
  end;
end.
```

BAB IX.
REKURSI

A. Konsep Rekursi

Rekursi adalah teknik pemrograman yang memungkinkan sebuah fungsi memanggil dirinya sendiri. Hal ini memungkinkan programmer untuk menulis program yang lebih ringkas dan mudah dipahami untuk menyelesaikan masalah yang kompleks. Contoh program pascalnya

```
function Faktorial(n: integer): integer;
begin
  if n = 0 then
  begin
    Faktorial := 1;
  end
  else
  begin
    Faktorial := n * Faktorial(n - 1);
  end;
end;
```

Pada contoh di atas, fungsi `Faktorial` menghitung faktorial dari bilangan `n` dengan menggunakan rekursi. Fungsi ini memanggil dirinya sendiri untuk menghitung faktorial dari bilangan `n-1`, dan kemudian mengalikan hasilnya dengan `n`.

B. Kompleksitas Waktu dan Ruang Rekursi

Kompleksitas waktu dan ruang rekursi adalah faktor penting yang perlu dipertimbangkan saat menggunakan rekursi. Kompleksitas waktu rekursi adalah waktu yang dibutuhkan untuk menjalankan fungsi rekursif. Kompleksitas ruang rekursi adalah memori yang dibutuhkan untuk menyimpan panggilan fungsi rekursif.

Secara umum, kompleksitas waktu rekursi untuk fungsi rekursif yang sederhana adalah O(n^k), di mana `n` adalah parameter input dan `k` adalah kedalaman rekursi. Kompleksitas ruang rekursi untuk fungsi rekursif yang sederhana adalah O(k).

C. Penerapan Rekursi

Rekursi dapat digunakan untuk menyelesaikan berbagai masalah pemrograman, seperti:

a. Menghitung faktorial: Seperti yang dicontohkan di atas, fungsi rekursif dapat digunakan untuk menghitung faktorial dari bilangan.
b. Mencari nilai Fibonacci: Nilai Fibonacci dapat dihitung dengan menggunakan fungsi rekursif.
c. Menjelajahi struktur data pohon: Struktur data pohon dapat dijelajahi dengan menggunakan teknik rekursi pre-order, in-order, dan post-order.
d. Memproses string secara rekursif: String dapat diproses secara rekursif untuk menyelesaikan berbagai tugas, seperti membalik string, mencari substring, dan memeriksa apakah string merupakan palindrome.

Rekursi adalah teknik pemrograman yang memungkinkan sebuah fungsi untuk memanggil dirinya sendiri. Teknik ini sering digunakan untuk menyelesaikan masalah yang dapat dipecahkan dengan cara membagi masalah menjadi sub-masalah yang lebih kecil dan serupa. Dalam fisika, rekursi dapat dimanfaatkan untuk berbagai perhitungan dan simulasi. Berikut adalah 5 contoh perhitungan fisika yang dapat diselesaikan dengan rekursi di bahasa Pascal:

1. Menghitung Trajektори Peluru dengan Persamaan Gerak Parabola:

```
program TrajektoriPeluru;

var
  kecepatanAwal: real;
  sudutAwal: real;
  gravitasi: real;
  waktu: real;
  x, y: real;

function PosisiX(waktu: real): real;
begin
  PosisiX := kecepatanAwal * waktu * Cos(sudutAwal);
end;

function PosisiY(waktu: real): real;
begin
  PosisiY := kecepatanAwal * waktu * Sin(sudutAwal) - 0.5 * gravitasi *
waktu ^ 2;
```

```
end;

begin
  Write('Masukkan kecepatan awal: ');
  ReadLn(kecepatanAwal);
  Write('Masukkan sudut awal (dalam derajat): ');
  ReadLn(sudutAwal);
  Write('Masukkan percepatan gravitasi: ');
  ReadLn(gravitasi);
  Write('Masukkan waktu: ');
  ReadLn(waktu);

  x := PosisiX(waktu);
  y := PosisiY(waktu);

  WriteLn('Posisi x pada waktu ', waktu, ': ', x);
  WriteLn('Posisi y pada waktu ', waktu, ': ', y);
end.
```

Penjelasan:

Program ini menghitung траектории peluru dengan menggunakan rekursi untuk menghitung posisi peluru (`x` dan `y`) pada waktu yang ditentukan. Fungsi `PosisiX` menghitung posisi horizontal peluru, sedangkan fungsi `PosisiY` menghitung posisi vertikal peluru. Fungsi-fungsi ini menggunakan rumus gerak parabola yang dimodifikasi untuk memperhitungkan percepatan gravitasi.

2. Menghitung Momentum Total Sistem Fisika dengan Rekursi:

```
program MomentumTotal;

var
  jumlahBenda: integer;
  massa: array[1..100] of real;
  kecepatan: array[1..100] of real;
  momentumTotal: real;

function MomentumBenda(i: integer): real;
```

```
begin
  MomentumBenda := massa[i] * kecepatan[i];
end;

begin
  Write('Masukkan jumlah benda: ');
  ReadLn(jumlahBenda);

  for i := 1 to jumlahBenda do
  begin
    Write('Masukkan massa benda ', i, ': ');
    ReadLn(massa[i]);
    Write('Masukkan kecepatan benda ', i, ': ');
    ReadLn(kecepatan[i]);
  end;

  momentumTotal := 0;
  for i := 1 to jumlahBenda do
  begin
    momentumTotal := momentumTotal + MomentumBenda(i);
  end;

  WriteLn('Momentum total sistem: ', momentumTotal);
end.
```

Penjelasan:

Program ini menghitung momentum total sistem fisika dengan rekursi. Fungsi `MomentumBenda` menghitung momentum individual dari setiap benda, dan kemudian fungsi ini dipanggil secara rekursif untuk menghitung momentum total dari semua benda dalam sistem.

3. Menyelesaikan Persamaan Diferensial Orde Pertama dengan Rekursi:

```
program PersamaanDiferensial;

var
  x0, y0: real;
  h: real;
```

```
  n: integer;
  y: array[0..100] of real;

function f(x, y: real): real;
begin
  f := x * y;  { Replace this function with your actual differential equation
}
end;

function EulerMethod(x0, y0, h, n, i: integer): real;
begin
  if i = 0 then
    EulerMethod := y0
  else
    EulerMethod := EulerMethod(x0, y0, h, n, i - 1) + h * f(x0 + (i - 1) * h,
EulerMethod(x0, y0, h, n, i - 1));
end;

begin
  Write('Masukkan nilai awal x: ');
  ReadLn(x0);
  Write('Masukkan nilai awal y: ');
  ReadLn(y0);
  Write('Masukkan step size (h): ');
  ReadLn(h);
  Write('Masukkan jumlah iterasi (n): ');
  ReadLn(n);

  for i := 0 to n do
  begin
    y[i] := EulerMethod(x0, y0, h, n, i);
  end;

  WriteLn('Nilai y untuk x dari ', x0, ' sampai ', x0 + n * h, ':');
  for i := 0 to n do
  begin
```

```
    WriteLn('x = ', x0 + i * h, ', y = ', y[i]);
  end;
end.
```

4. Menghitung Fraktal Sierpinski Triangle dengan Rekursi:

```
program FraktalSierpinskiTriangle;

var
  level: integer;
  ukuranSisi: real;
  x1, y1, x2, y2, x3, y3: real;

procedure SierpinskiTriangle(x1, y1, x2, y2, x3, y3, level: integer);
begin
  if level = 0 then
  begin
    DrawLine(x1, y1, x2, y2);
    DrawLine(x2, y2, x3, y3);
    DrawLine(x3, y3, x1, y1);
  end
  else
  begin
    SierpinskiTriangle((x1 + x2) / 2, (y1 + y2) / 2, x2, y2, (x2 + x3) / 2, (y2
+ y3) / 2, level - 1);
    SierpinskiTriangle(x1, y1, (x1 + x2) / 2, (y1 + y2) / 2, (x1 + x3) / 2, (y1
+ y3) / 2, level - 1);
    SierpinskiTriangle((x2 + x3) / 2, (y2 + y3) / 2, x3, y3, (x1 + x3) / 2, (y1
+ y3) / 2, level - 1);
  end;
end;

begin
  Write('Masukkan level fraktal: ');
  ReadLn(level);
  Write('Masukkan ukuran sisi segitiga: ');
```

```
  ReadLn(ukuranSisi);

  x1 := -ukuranSisi / 2;
  y1 := -sqrt(3) * ukuranSisi / 4;
  x2 := ukuranSisi / 2;
  y2 := -sqrt(3) * ukuranSisi / 4;
  x3 := 0;
  y3 := sqrt(3) * ukuranSisi / 4;

  SierpinskiTriangle(x1, y1, x2, y2, x3, y3, level);
end.
```

Penjelasan:

Program ini menggambar fraktal Sierpinski Triangle dengan menggunakan rekursi. Fungsi `SierpinskiTriangle` menggambar segitiga Sierpinski pada level tertentu. Pada level 0, fungsi ini menggambar segitiga dasar. Pada level selanjutnya, fungsi ini membagi segitiga menjadi 3 segitiga yang lebih kecil dan memanggil dirinya sendiri secara rekursif untuk menggambar segitiga-segitiga tersebut.

5. Mensimulasikan Gerak Brown dengan Rekursi:

```
program SimulasiGerakBrown;

var
  x, y: real;
  langkah: real;
  jumlahLangkah: integer;
  sudut: real;

function LangkahAcak(): real;
begin
  LangkahAcak := random * 2 * PI - PI;
end;

function GerakBrown(x0, y0, langkah: real, jumlahLangkah: integer):
array[0..jumlahLangkah] of real;
```

```
var
  i: integer;
begin
  GerakBrown[0] := [x0, y0];
  for i := 1 to jumlahLangkah do
  begin
    sudut := LangkahAcak();
    GerakBrown[i][0] := GerakBrown[i - 1][0] + langkah * Cos(sudut);
    GerakBrown[i][1] := GerakBrown[i - 1][1] + langkah * Sin(sudut);
  end;
end;

begin
  Write('Masukkan langkah awal (x, y): ');
  ReadLn(x, y);
  Write('Masukkan ukuran langkah: ');
  ReadLn(langkah);
  Write('Masukkan jumlah langkah: ');
  ReadLn(jumlahLangkah);

  var posisi := GerakBrown(x, y, langkah, jumlahLangkah);

  for i := 0 to jumlahLangkah do
  begin
    WriteLn(posisi[i][0], ', ', posisi[i][1]);
  end;
end.
```

Penjelasan:

Program ini mensimulasikan gerak Brown partikel dengan menggunakan rekursi. Fungsi LangkahAcak menghasilkan sudut acak untuk pergerakan partikel. Fungsi GerakBrown mensimulasikan pergerakan partikel dengan langkah acak dan menyimpan posisinya dalam array.

D. Rangkuman

Rekursi adalah teknik pemrograman yang memungkinkan sebuah fungsi memanggil dirinya sendiri. Rekursi dapat digunakan untuk menyelesaikan berbagai masalah pemrograman dengan cara yang lebih ringkas dan mudah dipahami. Namun, penting untuk mempertimbangkan kompleksitas waktu dan ruang rekursi saat menggunakan rekursi.

E. Test Formatif

1. Jelaskan apa itu rekursi dalam bahasa Pascal.
2. Bagaimana cara menulis program Pascal yang menggunakan rekursi?
3. Jelaskan apa itu kompleksitas waktu rekursi.
4. Berikan contoh penerapan rekursi untuk menyelesaikan masalah pemrograman.
6. Tuliskan program Pascal yang menggunakan rekursi untuk menghitung nilai Fibonacci.

F. Umpan Balik

a. Apakah materi bab ini sudah jelas dan mudah dipahami?
b. Apakah contoh-contoh program yang diberikan sudah cukup membantu?
c. Apakah ada bagian materi yang masih belum dipahami?

G. Tindak Lanjut

a. Cobalah untuk membuat program Pascal yang menggunakan rekursi untuk menyelesaikan masalah-masalah lain.
b. Carilah informasi lebih lanjut tentang rekursi dalam bahasa Pascal.
c. Diskusikan dengan teman atau dosen tentang penggunaan rekursi dalam pemrograman.

H. Kunci Jawaban Formatif

1. Rekursi adalah teknik pemrograman yang memungkinkan sebuah fungsi memanggil dirinya sendiri.
2. Cara menulis program Pascal yang menggunakan rekursi adalah dengan membuat fungsi yang memanggil dirinya sendiri dengan parameter yang berbeda.
3. Kompleksitas waktu rekursi adalah waktu yang dibutuhkan untuk menjalankan fungsi rekursif.
4. Contoh penerapan rekursi untuk menyelesaikan masalah pemrograman adalah menghitung faktorial, mencari nilai Fibonacci, menjelajahi struktur data pohon, dan memproses string secara rekursif.

5. Berikut adalah program Pascal yang menggunakan rekursi untuk menghitung nilai Fibonacci:

```
function Fibonacci(n: integer): integer;
begin
  if n = 0 then
  begin
    Fibonacci := 0;
  end
  else if n = 1 then
  begin
    Fibonacci := 1;
  end
  else
  begin
    Fibonacci := Fibonacci(n - 1) + Fibonacci(n - 2);
  end;
end;
```

BAB X.
FILE HANDLING

A. Konsep Dasar File Handling

File handling merupakan proses membaca, menulis, dan mengelola file dalam program. File adalah kumpulan data yang disimpan dalam media penyimpanan, seperti hard disk, flash drive, atau CD-ROM. File handling adalah komponen penting dalam pemrograman karena memungkinkan program untuk menyimpan dan mengakses data yang terstruktur.

B. Operasi File Handling

Operasi dasar file handling meliputi:

a. Membuka file: Membuka file berarti membuat koneksi antara program dan file yang ingin diakses.
b. Membaca file: Membaca file berarti mengambil data dari file ke dalam program.
c. Menulis file: Menulis file berarti menyimpan data dari program ke file.
d. Menutup file: Menutup file berarti mengakhiri koneksi antara program dan file.

C. Struktur Data File

Terdapat beberapa struktur data file yang umum digunakan, yaitu:

a. File teks: File teks berisi data yang dapat dibaca oleh manusia, biasanya dalam format ASCII.
b. File biner: File biner berisi data yang tidak dapat dibaca oleh manusia, biasanya dalam format byte.
c. File sekuensial: File sekuensial berisi data yang disimpan dalam urutan tertentu.
d. File terstruktur: File terstruktur berisi data yang diorganisir dalam struktur yang kompleks, seperti tabel atau hirarki.

D. Penanganan Kesalahan File Handling

Saat melakukan file handling, penting untuk menangani kesalahan yang mungkin terjadi, seperti file tidak ada, file tidak dapat dibuka, atau disk penuh. Penanganan kesalahan yang tepat dapat membantu program untuk tetap stabil dan berfungsi dengan baik. Contoh Implementasi File Handling. Berikut adalah contoh sederhana untuk membaca dan menulis file teks:

```
program ContohFileHandling;

var
  f: text;
  baris: string;

begin
  Assign(f, 'contoh.txt');
  Rewrite(f);

  WriteLn(f, 'Ini adalah baris pertama.');
  WriteLn(f, 'Ini adalah baris kedua.');

  Close(f);

  Assign(f, 'contoh.txt');
  Reset(f);

  while not Eof(f) do
  begin
    ReadLn(f, baris);
    WriteLn(baris);
  end;

  Close(f);
end.
```

Contoh program di atas menunjukkan cara membuka file teks, menulis dua baris data ke dalam file, dan kemudian membaca file dan menampilkan isinya.

```
program PhysicsCalculations;

var
  fInput: text;
  fOutput: text;
```

```
  data: string;
  mass: real;
  velocity: real;
  kineticEnergy: real;

begin
  Assign(fInput, 'physicsInput.txt');
  Reset(fInput);

  Assign(fOutput, 'physicsOutput.txt');
  Rewrite(fOutput);

  ReadLn(fInput, data);

  // Parse the input data into mass and velocity
  mass := StrToFloat(data);

  ReadLn(fInput, data);
  velocity := StrToFloat(data);

  // Calculate kinetic energy
  kineticEnergy := 0.5 * mass * Sqr(velocity);

  WriteLn(fOutput, 'Mass: ', mass);
  WriteLn(fOutput, 'Velocity: ', velocity);
  WriteLn(fOutput, 'Kinetic Energy: ', kineticEnergy);

  Close(fInput);
  Close(fOutput);
end.
```

Penjelasan Program Pascal untuk Perhitungan Fisika dengan File Handling. Program ini dirancang untuk menghitung energi kinetik suatu objek berdasarkan massa dan kecepatannya. Program ini memanfaatkan file handling untuk membaca data input dari file teks dan menulis hasil perhitungan ke file teks lain.

Langkah-langkah Program:

1. Membuka File Input dan Output:
   - `Assign(fInput, 'physicsInput.txt'); Reset(fInput);`: Baris ini membuka file `physicsInput.txt` untuk dibaca dan menetapkannya ke variabel `fInput`.
   - `Assign(fOutput, 'physicsOutput.txt'); Rewrite(fOutput);`: Baris ini membuka file `physicsOutput.txt` untuk ditulis dan menetapkannya ke variabel `fOutput`.
2. Membaca Data Input:
   - `ReadLn(fInput, data);`: Baris ini membaca satu baris teks dari file input ke dalam variabel `data`.
   - `mass := StrToFloat(data);`: Baris ini mengonversi string `data` menjadi angka real dan menyimpannya dalam variabel `mass`.
   - `ReadLn(fInput, data);`: Baris ini membaca baris teks lain dari file input ke dalam variabel `data`.
   - `velocity := StrToFloat(data);`: Baris ini mengonversi string `data` menjadi angka real dan menyimpannya dalam variabel `velocity`.
3. Menghitung Energi Kinetik:
   - `kineticEnergy := 0.5 * mass * Sqr(velocity);`: Baris ini menghitung energi kinetik menggunakan rumus `KE = 0.5 * mv^2`.
4. Menulis Hasil Perhitungan:
   - `WriteLn(fOutput, 'Massa: ', mass);`: Baris ini menulis nilai massa ke file output bersama dengan label.
   - `WriteLn(fOutput, 'Kecepatan: ', velocity);`: Baris ini menulis nilai kecepatan ke file output bersama dengan label.
   - `WriteLn(fOutput, 'Energi Kinetik: ', kineticEnergy);`: Baris ini menulis nilai energi kinetik yang dihitung ke file output bersama dengan label.
5. Menutup File Input dan Output:
   - `Close(fInput);`: Baris ini menutup file input.
   - `Close(fOutput);`: Baris ini menutup file output.

Cara Menggunakan Program:

1. Buat Dua File Teks:
   - `physicsInput.txt`: Letakkan nilai massa dan kecepatan yang ingin Anda hitung energinya pada baris terpisah dalam file ini.
   - `physicsOutput.txt`: File ini akan dibuat oleh program dan akan berisi nilai energi kinetik yang dihitung.

2. Simpan Program Pascal:
   - Simpan kode program sebagai file `.pas`, seperti `physics Calculations.pas`.
3. Kompilasi dan Jalankan Program:
   - Gunakan kompilator atau IDE Pascal untuk mengompilasi file `.pas` menjadi file yang dapat dieksekusi.
   - Jalankan file yang dapat dieksekusi, dan program akan membaca data input dari `physicsInput.txt`, menghitung energi kinetik, dan menulis hasilnya ke `physicsOutput.txt`.

Contoh Data Input dalam `physicsInput.txt`:

```
10.0
20.0
```

Output yang Sesuai dalam `physicsOutput.txt`:

```
Massa: 10.0
Kecepatan: 20.0
Energi Kinetik: 200.0
```

Penjelasan Tambahan:

- Program ini menggunakan fungsi `StrToFloat` untuk mengonversi string data input menjadi angka real.
- Fungsi `Sqr` digunakan untuk menghitung kuadrat dari suatu nilai.
- Program ini dapat dimodifikasi untuk menghitung jenis perhitungan fisika lainnya dengan mengubah rumus dan data input yang sesuai.

E. Rangkuman

Bab ini membahas tentang konsep dasar file handling dalam pemrograman, termasuk operasi membaca dan menulis file, serta penggunaan berbagai struktur data file. Pemahaman tentang file handling sangat penting bagi programmer untuk dapat menyimpan, mengelola, dan memproses data secara efisien.

F. Test Formatif

1. Apa yang dimaksud dengan file handling?
2. Sebutkan operasi dasar file handling.
3. Jelaskan apa yang dimaksud dengan file teks, file biner, dan file sekuensial.

4. Bagaimana cara menangani kesalahan yang terjadi saat melakukan file handling?
5. Berikan contoh implementasi file handling untuk membaca dan menulis file teks.

G. Umpan Balik

Mahasiswa dapat memberikan umpan balik mengenai materi bab ini melalui forum diskusi online atau langsung kepada dosen.

H. Tindak Lanjut

Mahasiswa dapat mempelajari lebih lanjut tentang file handling dengan membaca buku, artikel, dan tutorial online. Mahasiswa juga dapat mencoba menerapkan file handling dalam proyek pemrograman mereka sendiri.

I. Kunci Jawaban Formatif

1. File handling adalah proses membaca, menulis, dan mengelola file dalam program.
2. Operasi dasar file handling meliputi membuka file, membaca file, menulis file, dan menutup file.
3. File teks berisi data yang dapat dibaca oleh manusia, biasanya dalam format ASCII. File biner berisi data yang tidak dapat dibaca oleh manusia, biasanya dalam format byte. File sekuensial berisi data yang disimpan dalam urutan tertentu.
4. Penanganan kesalahan file handling dapat dilakukan dengan menggunakan try-except block atau dengan menggunakan pernyataan if untuk memeriksa kondisi error.
2. Contoh implementasi file handling untuk membaca dan menulis file teks dapat dilihat pada contoh kode yang diberikan sebelumnya.

BAB XI.
STRUKTUR DATA LANJUTAN

A. Daftar Berantai (Linked List)

Daftar berantai adalah struktur data yang terdiri dari elemen-elemen yang disebut node. Setiap node memiliki data dan referensi ke node berikutnya dalam daftar. Daftar berantai dapat digunakan untuk menyimpan data secara dinamis dan efisien, dan memungkinkan operasi penyisipan, penghapusan, dan pencarian data dengan mudah.

B. Tumpukan (Stack)

Tumpukan adalah struktur data LIFO (Last In, First Out). Elemen yang ditambahkan terakhir ke tumpukan adalah elemen pertama yang akan dihapus. Tumpukan sering digunakan untuk menyimpan data sementara, seperti panggilan fungsi dan operasi undo/redo.

C. Antrean (Queue)

Antrean adalah struktur data FIFO (First In, First Out). Elemen yang ditambahkan pertama ke antrean adalah elemen pertama yang akan dihapus. Antrean sering digunakan untuk menyimpan data yang harus diproses secara berurutan, seperti antrian pelanggan di toko.

D. Pohon (Tree)

Pohon adalah struktur data hierarkis yang terdiri dari node yang disebut akar (root) dan node-node lain yang disebut anak (child). Setiap node dapat memiliki beberapa anak. Pohon sering digunakan untuk menyimpan data yang memiliki hubungan hierarkis, seperti struktur organisasi perusahaan.

E. Graf (Graph)

Graf adalah struktur data yang terdiri dari node-node (vertex) dan hubungan antar node yang disebut sisi (edge). Sisi dapat berarah (directed) atau tidak berarah (undirected). Graf sering digunakan untuk memodelkan hubungan antar entitas, seperti jaringan sosial atau peta jalan.

F. Struktur data lanjut dapat diterapkan dalam berbagai program, seperti:

a. Kompiler: Daftar berantai dan tumpukan digunakan untuk menyimpan dan mengelola token dan simbol selama proses kompilasi.
b. Sistem operasi: Antrean digunakan untuk mengelola proses dan perangkat I/O.

c. Basis data: Pohon dan graf digunakan untuk menyimpan dan mengelola data
d. Jaringan komputer: Graf digunakan untuk memodelkan jaringan dan merutekan data.
e. Algoritma pencarian: Pohon dan graf digunakan untuk mengimplementasikan algoritma pencarian yang efisien, seperti pohon B dan algoritma Dijkstra.
f. Permainan dan simulasi: Struktur data lanjut dapat digunakan untuk memodelkan dunia game dan simulasi, seperti peta permainan, karakter, dan interaksi antar elemen.

G. Pemilihan Struktur Data yang Tepat

Pemilihan struktur data yang tepat untuk suatu aplikasi tergantung pada beberapa faktor, antara lain:

a. Jenis data: Struktur data yang berbeda cocok untuk jenis data yang berbeda, misalnya daftar berantai untuk data string, tumpukan untuk data sementara, dan pohon untuk data hierarkis.
b. Operasi yang diperlukan: Struktur data tertentu lebih efisien untuk operasi tertentu, misalnya antrean untuk operasi FIFO dan pohon untuk pencarian data.
c. Kinerja: Struktur data yang berbeda memiliki performa yang berbeda, misalnya daftar berantai umumnya lebih lambat daripada array untuk akses data acak.

Berikut adalah contoh implementasi sederhana daftar berantai dalam Pascal:

```
type
  TNode = record
    data: string;
    next: ^TNode;
  end;

var
  head: ^TNode;

begin
  // Menambahkan elemen ke daftar
  head := New(TNode);
```

```
  head^.data := 'Data pertama';
  head^.next := nil;

  head := New(TNode);
  head^.data := 'Data kedua';
  head^.next := head;

  // Menampilkan data dalam daftar
  while head <> nil do
  begin
    WriteLn(head^.data);
    head := head^.next;
  end;

  // Menghapus elemen dari daftar
  head := head^.next;
  Dispose(head^.prev);
end.
```

Contoh kode di atas menunjukkan cara membuat daftar berantai sederhana, menambahkan elemen ke daftar, menampilkan data dalam daftar, dan menghapus elemen dari daftar.

H. Rangkuman

Bab ini membahas tentang struktur data lanjut dalam pemrograman, termasuk daftar berantai, tumpukan, antrean, pohon, dan graf. Struktur data lanjut ini memiliki karakteristik dan kegunaannya masing-masing yang penting untuk dipahami dan diterapkan dalam berbagai program yang kompleks.

I. Test Formatif

1. Apa yang dimaksud dengan daftar berantai?
2. Bagaimana cara kerja tumpukan?
3. Apa perbedaan antara antrean dan tumpukan?
4. Bagaimana cara menggunakan pohon untuk menyimpan data hierarkis?
5. Apa saja penerapan graf dalam pemrograman?

6. Faktor apa yang perlu dipertimbangkan saat memilih struktur data yang tepat?
7. Berikan contoh implementasi sederhana daftar berantai dalam bahasa pemrograman yang Anda pilih.

J. Umpan Balik

Mahasiswa dapat memberikan umpan balik mengenai materi bab ini melalui forum diskusi online atau langsung kepada dosen.

K. Tindak Lanjut

Mahasiswa dapat mempelajari lebih lanjut tentang struktur data lanjut dengan membaca buku, artikel, dan tutorial online. Mahasiswa juga dapat mencoba menerapkan struktur data lanjut dalam proyek pemrograman mereka sendiri.

L. Kunci Jawaban Formatif

1. Daftar berantai adalah struktur data yang terdiri dari elemen-elemen yang disebut node. Setiap node memiliki data dan referensi ke node berikutnya dalam daftar.
2. Tumpukan adalah struktur data LIFO (Last In, First Out). Elemen yang ditambahkan terakhir ke tumpukan adalah elemen pertama yang akan dihapus.
3. Antrean adalah struktur data FIFO (First In, First Out). Elemen yang ditambahkan pertama ke antrean adalah elemen pertama yang akan dihapus.
4. Pohon dapat digunakan untuk menyimpan data hierarkis dengan menghubungkan node-node yang memiliki hubungan hierarkis.
5. Graf dapat digunakan untuk memodelkan hubungan antar entitas, seperti jaringan sosial atau peta jalan.
6. Faktor-faktor yang perlu dipertimbangkan saat memilih struktur data yang tepat adalah jenis data, operasi yang diperlukan, dan kinerja.
7. Contoh implementasi sederhana daftar berantai dalam Pascal dapat dilihat pada contoh kode yang diberikan sebelumnya.