



*php*

# BONGKAR RAHASIA PEMROGRAMAN

PANDUAN ASYIK BELAJAR PEMROGRAMAN DASAR DENGAN BAHASA  
PEMROGRAMAN PHP UNTUK SEMUA USIA



Yusup Indra Wijaya, S.Kom., M.Kom

# **BONGKAR RAHASIA PEMROGRAMAN, PANDUAN ASYIK BELAJAR PEMROGRAMAN DASAR DENGAN BAHASA PEMROGRAMAN PHP UNTUK SEMUA USIA**

## **Penulis :**

Yusup Indra Wijaya, S.Kom., M.Kom.

## **Editor :**

Antoni Pardede, S.Si., M.Si., Ph.D

## **Tata Letak :**

Aris Setia Noor, S.E., M.Si

## **Desain Sampul:**

M. Fikri Ansari, S.Kom

## **Penerbit :**

Universitas Islam Kalimantan Muhammad Arsyad Al-Banjari  
Banjarmasin

## **Redaksi :**

Gedung A UPT Publikasi dan Pengelolaan Jurnal Universitas Islam  
Kalimantan Muhammad  
Arsyad Al-Banjary  
Jl. Adhyaksa No. 2 Kayutangi Banjarmasin, Kalimantan Selatan  
Telepon (faks) : 0511 – 3304352

**Cetakan Pertama : Januari 2024**

**ISBN : 978-623-8189-13-7**

ISBN 978-623-8189-13-7



Hak Cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa izin tertulis dari Penerbit. Isi diluar tanggung jawab penerbit

# KATA PENGANTAR

Puji Syukur Kehadirat Allah subhanahu wa ta'ala, karena atas izin dan kehendaknyalah buku Bongkar Rahasia Pemrograman, Panduan Asyik Belajar Pemrograman Dasar Dengan Bahasa Pemrograman PHP Untuk Semua Usia ini dapat diselesaikan.

Kami berharap Buku ini dapat menjadi rujukan bagi semua yang ingin belajar pemrograman dengan mudah, asyik dan tidak membosankan bagi semua usia.

Kepada semua pihak yang telah membantu penerbitan dan publikasi diucapkan banyak terima kasih dan penghargaan yang setinggi-tingginya. Semoga buku ini bermanfaat dan berguna bagi para pengguna.

Banjarmasin, Januari 2024

Penulis,

# DAFTAR ISI

DAFTAR ISI.....	iv
DAFTAR GAMBAR .....	vi
BAB 1 PENGENALAN PEMROGRAMAN.....	1
Sub-Bab 1.1 Apa itu Pemrograman? .....	1
Sub-Bab 1.2 Mengapa Belajar Pemrograman?.....	5
Sub-Bab 1.3 Jenis-Jenis Bahasa Pemrograman .....	8
Sub-Bab 1.4 Lingkungan Pengembangan Pemrograman .....	11
BAB 2 DASAR-DASAR PEMROGRAMAN .....	14
Sub-Bab 2.1 Hello World, Program Pertama Kamu. 14	
Sub-bab 2.2 Variabel dan Tipe Data.....	20
Sub-bab 2.3 Operator dan Ekspresi .....	26
Sub-bab 2.4 Struktur Kontrol Percabangan .....	33
Sub-bab 2.5 Struktur Kontrol Perulangan.....	39
BAB 3 STRUKTUR DATA DASAR.....	46
Sub-bab 3.1 Array dan List.....	46
Sub-bab 3.2 String dan Teks Processing.....	51
Sub-bab 3.3 Tabel dan Matriks .....	56
Sub-bab 3.4 Penggunaan Data Structures .....	62
BAB 4 FUNGSI DAN PROSEDUR .....	68
Sub-bab 4.1 Apa itu Fungsi?.....	68
Sub-bab 4.2 Membuat dan Memanggil Fungsi.....	74
Sub-bab 4.3 Parameter dan Argumen .....	80
Sub-bab 4.4 Rekursi (Pemanggilan Fungsi Diri Sendiri).....	84
BAB 5 OBJECT-ORIENTED PROGRAMMING (PEMROGRAMAN BERORIENTASI OBJEK) .....	87
Sub-bab 5.1 Konsep Dasar OOP.....	87

Sub-bab 5.2 Class dan Objek .....	90
Sub-bab 5.3 Enkapsulasi dan Abstraksi.....	91
Sub-bab 5.4 Pewarisan dan Polimorfisme .....	94
<b>BAB 6 PENANGANAN ERROR DAN DEBUGGING</b> .....	<b>98</b>
Sub-bab 6.1 Error dan Exception.....	98
Sub-bab 6.2 Handling Exceptions.....	103
Sub-bab 6.3 Debugging Tools dan Teknik .....	108
<b>BAB 7 PEMROGRAMAN BERBASIS KASUS.....</b>	<b>115</b>
Sub-bab 7.1 Studi Kasus I Pembuatan Aplikasi Sederhana.....	115
Sub-bab 7.2 Studi Kasus II Pengembangan Game Sederhana.....	119
<b>DAFTAR PUSTAKA.....</b>	<b>123</b>

## DAFTAR GAMBAR

Gambar 1 Ilustrasi Variabel (Sumber dari ankamunesia.com) .....	4
Gambar 2 HTML "Hello, World!" .....	15
Gambar 3 Hasil "Hello, World!" HTML .....	15
Gambar 4 PHP "Hello, World!" .....	17
Gambar 5 Hasil "Hello, World!" PHP .....	17
Gambar 6 Variabel PHP.....	21
Gambar 7 Variabel pada HTML dan PHP .....	22
Gambar 8 Penggunaan Operator Aritmatika.....	27
Gambar 9 Operator Perbandingan .....	28
Gambar 10 Operator Logika .....	29
Gambar 11 Ekspresi.....	30
Gambar 12 Beberapa item .....	31
Gambar 13 Operator Perbandingan .....	32
Gambar 14 Penggunaan If Statement .....	34
Gambar 15 Penggunaan Else Statement .....	35
Gambar 16 Penggunaan Else Statement .....	36
Gambar 17 Penggunaan Else If.....	37
Gambar 18 Penggunaan Else If.....	38
Gambar 19 Penggunaan For Loop .....	41
Gambar 20 Penggunaan While Loop .....	42
Gambar 21 Penggunaan Do While .....	44
Gambar 22 Penggunaan Foreach .....	45
Gambar 23 Contoh Array untuk Buah .....	46
Gambar 24 Mengakses Array untuk Buah.....	47
Gambar 25 Array dan List .....	48
Gambar 26 Array Asosiatif.....	49
Gambar 27 String.....	51
Gambar 28 Menggabungkan String.....	52
Gambar 29 Menghitung Panjang String .....	52

Gambar 30 Membagi String.....	52
Gambar 31 Mengganti Teks dalam String .....	53
Gambar 32 Penggabungan String .....	54
Gambar 33 Pencarian Teks String .....	54
Gambar 34 Pemotongan String.....	54
Gambar 35 Mengganti Teks dalam String .....	55
Gambar 36 Konversi dalam String .....	55
Gambar 37 Tabel HTML .....	57
Gambar 38 Matrix.....	58
Gambar 39 Menambahkan Baris ke Tabel HTML .....	60
Gambar 40 Menambahkan Baris Baru ke Matrik.....	61
Gambar 41 Membuat Array Asosiatif dalam PHP .....	63
Gambar 42 Objek.....	64
Gambar 43 Fungsi.....	69
Gambar 44 Variabel dalam Fungsi .....	77
Gambar 45 Parameter lebih dari Satu .....	81
Gambar 46 Rekursi untuk Menghitung Faktorial .....	85
Gambar 47 Rekursi untuk Penghitung Mundur .....	86
Gambar 48 Enkapsulasi .....	92
Gambar 49 Abstraksi.....	94
Gambar 50 Polimorfisme.....	96
Gambar 51 Exception .....	100

# BAB 1

## PENGENALAN PEMROGRAMAN

### Sub-Bab 1.1 Apa itu Pemrograman?

Hai semua! Mari kita mulai petualangan kita dalam dunia pemrograman dengan pertanyaan sederhana: Apa sih pemrograman itu?

Pemrograman sebenarnya mirip dengan memberikan instruksi kepada komputer. Bayangkan jika komputer adalah teman kita yang pintar. Ketika kita ingin komputer melakukan sesuatu, kita memberi teman kita itu serangkaian instruksi yang sangat spesifik. Pikirkan saja ini seperti membuat daftar tugas untuk temanmu. Misalnya, kita ingin teman kita menggambar gambar burung. Instruksi kita mungkin terdengar seperti ini:

1. Ambil selembar kertas.
2. Gambar lingkaran untuk tubuh burung.
3. Tambahkan kepala dengan paruh.
4. Lukis sayap di kedua sisi tubuh.
5. Jangan lupa ekor dan mata.



Komputer kita adalah teman yang sangat patuh. Dia akan mengikuti instruksi kita dengan cermat dan menggambar burung sesuai dengan apa yang kita minta. Ini adalah dasar dari pemrograman - memberikan komputer instruksi yang sangat rinci. Namun, ada satu hal yang perlu diingat: komputer tidak bisa berpikir sendiri. Dia hanya bisa melakukan apa yang kita instruksikan. Jadi, pemrograman melibatkan pemikiran kreatif kita untuk merancang instruksi-instruksi ini sehingga komputer bisa melakukan hal-hal hebat.

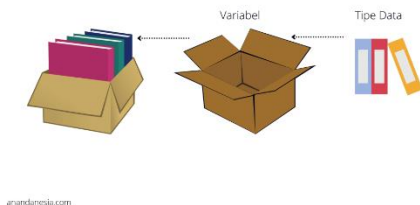
Pemrograman adalah cara kita berbicara dengan komputer, memberi tahu mereka apa yang harus dilakukan. Ini adalah kekuatan untuk menciptakan game seru, aplikasi pintar, dan bahkan membuat dunia maya di komputer. Dan yang terbaik dari semuanya adalah belajar pemrograman itu sangat menyenangkan! Jadi, mari bersiap-siap untuk menjelajahi dunia yang penuh dengan kode dan logika. Pemrograman adalah alat yang kuat untuk mengubah ide-ide kita menjadi kenyataan di dunia digital. Siap? Mari kita mulai!

Sekarang kita tahu bahwa pemrograman adalah cara kita berbicara dengan komputer, tapi bagaimana kita melakukannya?

Pemrograman itu mirip dengan bahasa rahasia. Jika kita ingin teman kita mengerti, kita harus berbicara dalam bahasa yang komputer mengerti. Dan, tahukah kamu apa bahasa itu? Itu disebut "kode". Kode adalah serangkaian instruksi yang ditulis dalam bahasa tertentu yang bisa dimengerti oleh komputer. Bahasa ini disebut "bahasa pemrograman." Tidak ada yang tahu berapa banyak bahasa pemrograman yang ada, tapi ada banyak pilihan! Ada bahasa seperti PHP, Python, Java, JavaScript, dan banyak lagi. Masing-masing memiliki aturan dan cara unik untuk berbicara dengan komputer.

Bagian hebatnya adalah, tidak ada batasan usia untuk memulai pemrograman. Baik kamu masih anak-anak, remaja, atau dewasa, kamu bisa belajar pemrograman. Itu seperti belajar bahasa asing, tapi bahasa ini digunakan untuk berbicara dengan komputer, bukan orang.

Saat kita belajar bahasa pemrograman, kita akan belajar tentang konsep-konsep dasar, seperti variabel (yang seperti kotak untuk menyimpan informasi), perulangan (saat kita mengulangi sesuatu), dan percabangan (ketika kita membuat keputusan). Ini adalah dasar dari hampir semua bahasa pemrograman.



*Gambar 1 Ilustrasi Variabel (Sumber dari ankamunesia.com)*

Ketika kita menulis kode, kita akan membuat program. Program ini adalah kumpulan instruksi yang membantu komputer melakukan tugas-tugas keren. Misalnya, kita bisa membuat program untuk permainan video, program untuk menghitung angka, atau bahkan program untuk membuat musik! Tentu saja, kita akan perlu banyak latihan dan eksperimen. Pemrograman adalah tentang mencoba hal-hal baru, membuat kesalahan, dan memperbaiki mereka. Tidak masalah jika

kamu belum tahu banyak, yang penting adalah selalu berani mencoba.

Jadi, mari kita bersiap-siap untuk menjelajahi dunia pemrograman. Ini adalah dunia di mana kita bisa menjadi pencipta, penemuan hal-hal baru, dan berpetualang dalam dunia digital. Ini adalah dunia yang seru dan penuh peluang. Selamat datang di dunia pemrograman!

## **Sub-Bab 1.2 Mengapa Belajar Pemrograman?**

Hai semuanya! Sekarang kita tahu apa itu pemrograman, tapi mengapa kita harus belajar pemrograman? Apa yang membuatnya begitu keren?

Pertama-tama, pemrograman adalah seperti memiliki superpower. Ini memberi kita kekuatan untuk menciptakan hal-hal yang belum pernah ada sebelumnya. Bayangkan bisa membuat permainan video sendiri atau aplikasi pintar yang membantu orang lain. Dengan pemrograman, kita bisa menjadi pencipta! Selain itu, pemrograman mengajarkan kita berpikir logis. Ini seperti

memecahkan teka-teki besar. Ketika kita menulis kode, kita harus merancang langkah-langkah dengan sangat hati-hati. Itu mengasah kemampuan kita untuk memecahkan masalah, tidak hanya di dunia pemrograman, tapi dalam kehidupan sehari-hari juga.

Pemrograman juga membuka pintu ke berbagai karir. Banyak pekerjaan di dunia ini membutuhkan orang yang tahu tentang pemrograman. Jadi, jika kamu suka pemrograman, kamu bisa memiliki pekerjaan yang keren di masa depan. Kamu juga bisa mengejar hobi yang seru dengan pemrograman. Misalnya, kamu bisa membuat situs web untuk berbagi cerita atau menggambar game kecil untuk teman-temanmu. Pemrograman memberi kita alat untuk mengubah ide-ide kita menjadi kenyataan. Hal yang hebat tentang pemrograman adalah bahwa tidak ada batasan usia. Kamu bisa mulai belajar pemrograman sejak dini. Bahkan, banyak anak-anak di seluruh dunia sudah menjadi ahli dalam pemrograman!

Belajar pemrograman juga bisa sangat menyenangkan. Saat kamu menulis kode, kamu bisa membuat karakter bergerak di layar, membuat suara, atau

bahkan mengendalikan robot. Ini adalah seperti bermain, tapi kamu juga belajar sekaligus! Jadi, alasan untuk belajar pemrograman sangat banyak. Ini adalah keterampilan seru yang membawa banyak manfaat. Mulailah hari ini, dan siapa tahu, kamu mungkin akan menjadi pembuat game atau programmer hkamul di masa depan. Selain itu, pemrograman juga membuka pintu bagi kita untuk mengeksplorasi teknologi. Dunia kita semakin digital, dan pemrograman adalah kunci untuk memahami dan mengontrol teknologi ini. Dengan pemrograman, kita bisa membuat teknologi bekerja untuk kita.

Pemrograman juga memberi kita kebebasan untuk menjadi inovatif. Saat kita belajar pemrograman, kita dapat mengembangkan ide-ide kreatif kita sendiri. Mungkin kamu punya ide untuk membuat aplikasi yang memudahkan orang dalam kehidupan sehari-hari, atau kamu ingin membuat permainan yang unik dan menyenangkan. Pemrograman memberi kita alat untuk mewujudkan ide-ide ini. Pemrograman juga memperluas cara kita berpikir. Saat kita menulis kode, kita harus memecah masalah menjadi langkah-langkah kecil. Ini

membantu kita berpikir lebih sistematis dan logis dalam menghadapi tantangan dalam hidup. Selain itu, pemrograman adalah cara yang bagus untuk memahami dunia di sekitar kita. Kita bisa membuat program untuk mengumpulkan dan menganalisis data, misalnya data cuaca atau data kesehatan. Ini bisa membantu kita memahami tren dan membuat keputusan yang lebih baik.

Jadi, kenapa kamu harus belajar pemrograman? Karena pemrograman memberi kita keterampilan super, membuka peluang pekerjaan yang menarik, memungkinkan kita berkreasi, dan membuat kita berpikir lebih pintar. Dan yang terpenting, belajar pemrograman itu sangat seru!

### **Sub-Bab 1.3 Jenis-Jenis Bahasa Pemrograman**

Hai, sobat programmer! Sekarang kita akan berbicara tentang sesuatu yang seru: bahasa pemrograman. Bayangkan saja, ini adalah bahasa rahasia yang digunakan untuk berbicara dengan komputer. Mari kita lihat beberapa bahasa pemrograman yang keren:

1. Python, Python adalah bahasa pemrograman yang cocok untuk pemula. Ini seperti bahasa Inggris sederhana. Kamu bisa menggunakannya untuk membuat game, situs web, dan bahkan robot! Iya, robot!
2. JavaScript, Jika kamu ingin membuat situs web yang interaktif, JavaScript adalah jawabannya. Kamu bisa membuat tombol yang berkedip, gambar yang bergerak, dan banyak lagi. Ini seperti memberi kehidupan pada situs webmu.
3. Java, Ini bukan tentang kopi, melainkan tentang aplikasi. Java adalah bahasa yang digunakan untuk membuat aplikasi Android. Kamu bisa menciptakan aplikasi untuk ponsel pintarmu sendiri.
4. C++, Jika kamu ingin menjadi ahli dalam pembuatan game video, C++ adalah bahasa yang dipakai. Ini digunakan untuk menciptakan game-game seru yang kamu mainkan di komputer atau konsol.
5. Scratch, Ini adalah bahasa yang menyenangkan untuk teman-teman yang masih muda. Kamu bisa membuat



proyek sederhana dengan cara menata blok-blok kode seperti menyelesaikan teka-teki.

6. R, Jika kamu suka matematika dan angka, R adalah pilihan yang bagus. Ini digunakan untuk menganalisis data dan menemukan informasi menarik dari angka-angka tersebut.

7. PHP: PHP adalah bahasa pemrograman yang digunakan untuk mengembangkan situs web dinamis. Dengan PHP, kamu bisa membuat situs web yang berinteraksi dengan basis data dan pengguna.

Ada banyak bahasa pemrograman lainnya juga. Setiap bahasa memiliki ciri khas dan kegunaannya masing-masing. Jadi, saat kamu belajar pemrograman, pilihlah bahasa yang paling seru dan sesuai dengan minatmu. Penting untuk diingat bahwa belajar bahasa pemrograman itu seperti belajar bahasa asing. Kamu tidak perlu tahu semuanya sekaligus. Cobalah beberapa bahasa dan pilih yang paling seru bagimu. Selamat belajar, penjelajah dunia pemrograman!

## **Sub-Bab 1.4 Lingkungan Pengembangan Pemrograman**

Hai semuanya! Sekarang kita akan memasuki dunia lingkungan pengembangan pemrograman, yang sering disebut IDE (Integrated Development Environment). IDE adalah tempat di mana para pemrogram menciptakan perangkat lunak ajaib mereka. Ini seperti tempat bermain bagi para programmer!

1. Editor Teks : Ini adalah cara termudah untuk mulai. Kamu bisa menggunakan editor teks biasa seperti Notepad (untuk Windows) atau TextEdit (untuk Mac). Tapi jika kamu suka warna-warni dan bantuan tambahan, ada editor teks yang dioptimalkan untuk pemrograman seperti Visual Studio Code dan Sublime Text. Mereka akan memberikanmu fitur keren seperti penyorotan sintaks dan otomatisasi tertentu.

2. IDE Bahasa Pemrograman : Setiap bahasa pemrograman punya IDE khusus. Misalnya, jika kamu memprogram dalam Java, kamu bisa menggunakan

Eclipse atau IntelliJ IDEA. IDE ini memiliki semua alat yang kamu butuhkan untuk mengembangkan aplikasi dalam bahasa tertentu.

3. Platform Online : Ada platform online seperti Replit dan CodePen yang memungkinkanmu untuk menulis dan menjalankan kode langsung di browser. Mereka bagus jika kamu ingin berbagi proyekmu dengan teman atau ingin berlatih di mana saja.

4. Pengembangan Web : Jika kamu ingin menjadi pengembang web, kamu akan bekerja dengan HTML, CSS, dan JavaScript. Untuk itu, kamu hanya butuh teks editor dan browser untuk melihat hasilnya.

5. Jupyter Notebook : Jupyter Notebook adalah alat yang keren jika kamu ingin melakukan pemrograman interaktif, terutama dalam bahasa seperti Python. Ini memungkinkanmu menjalankan kode secara bertahap dan melihat hasilnya dalam langkah-langkah.

6. Mobile Development : Jika kamu ingin mengembangkan aplikasi mobile, seperti untuk Android,

kamu akan menggunakan Android Studio atau Xcode (untuk iOS).

7. Tools Khusus : Beberapa proyek pemrograman memiliki alat khusus. Misalnya, Unity untuk pembuatan game 3D, Arduino IDE untuk pemrograman perangkat keras, dan sebagainya.

Pilih lingkungan yang paling cocok untukmu berdasarkan bahasa pemrograman dan jenis proyek yang ingin kamu kerjakan. Jangan khawatir, kamu bisa mencoba beberapa IDE dan melihat mana yang paling kamu suka. Semua pemrogram pemula mulai dari sini, jadi jangan ragu untuk menjajal berbagai lingkungan pengembangan. Selamat mencoba dan selamat berkarya, teman-teman programmer!

## **BAB 2**

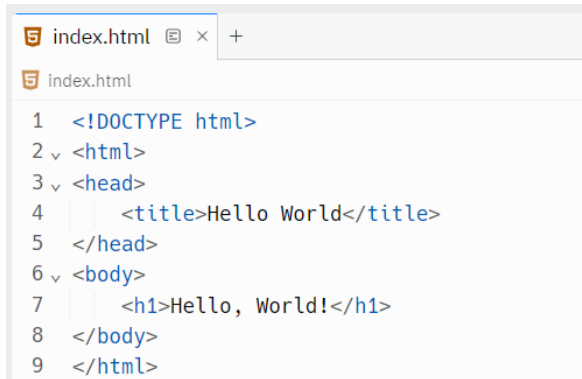
### **DASAR-DASAR PEMROGRAMAN**

#### **Sub-Bab 2.1 Hello World, Program Pertama Kamu**

Halo, semua orang yang ingin memulai petualangan pemrograman! Kami akan menjelaskan apa itu HTML, PHP, dan mengapa kita menggunakannya. Jangan khawatir, itu lebih sederhana daripada yang terlihat!

HTML (HyperText Markup Language) adalah bahasa yang digunakan untuk membuat halaman web. Ini seperti kerangka dasar yang mengatur bagaimana halaman web kita akan terlihat dan berperilaku. Dalam bahasa yang lebih sederhana, HTML adalah apa yang membuat teks, gambar, dan tautan muncul di situs web. Ini adalah fondasi dari web.

Contoh sederhana program "Hello World" menggunakan HTML:



```
index.html x +
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Hello World</title>
5 </head>
6 <body>
7   <h1>Hello, World!</h1>
8 </body>
9 </html>
```

Gambar 2 HTML "Hello, World!"

Maka akan tampil di browser seperti berikut :



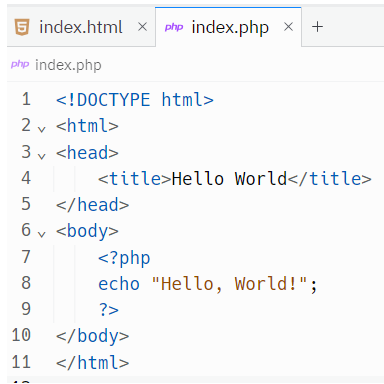
Gambar 3 Hasil "Hello, World!" HTML

Jadi, bagaimana itu bekerja? HTML digunakan untuk membuat struktur halaman web. Di atas, kita memiliki elemen-elemen seperti `<html>`, `<head>`, `<title>`, `<body>`, dan `<h1>`. Mereka digunakan untuk mengatur struktur dan konten halaman web. Hasilnya adalah

halaman web sederhana dengan teks "Hello, World!" yang ditampilkan dalam heading 1.

PHP (Hypertext Preprocessor) adalah bahasa pemrograman yang digunakan untuk membuat halaman web yang dinamis. Ini memungkinkan kita untuk berinteraksi dengan basis data, mengirim formulir, dan melakukan banyak hal lainnya yang membuat situs web kita lebih interaktif.

Contoh sederhana program "Hello World" menggunakan PHP:



```
index.html x  php index.php x +
php index.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Hello World</title>
5 </head>
6 <body>
7   <?php
8     echo "Hello, World!";
9   ?>
10 </body>
11 </html>
..
```

*Gambar 4 PHP "Hello, World!"*

Maka akan tampil di browser seperti berikut :



*Gambar 5 Hasil "Hello, World!" PHP*

Di sini, kita menggabungkan HTML dengan PHP. Ketika server web mengeksekusi kode PHP, itu akan menghasilkan output yang dapat ditampilkan di halaman HTML. Hasilnya adalah halaman web yang sama dengan pesan "Hello, World!" Jadi, kita menggunakan HTML untuk membangun struktur dasar halaman web dan PHP



untuk membuatnya dinamis. Ini adalah kombinasi yang hebat yang memungkinkan kita untuk membuat situs web yang mengesankan.

Sekarang, giliranmu untuk mencoba! Buka editor teks favoritmu, tulis kode HTML dan PHP "Hello World," dan lihat hasilnya. Selamat belajar dan selamat memulai perjalananmu dalam dunia pemrograman web yang menarik!

Mari kita perjelas mengapa kita menggunakan HTML dan PHP:

HTML (HyperText Markup Language) :

- Kerangka Dasar : HTML adalah kerangka dasar untuk setiap halaman web. Ini memberi tahu browser web bagaimana mengatur dan menampilkan konten.
- Teks, Gambar, dan Tautan : Dengan HTML, kita dapat menambahkan teks, gambar, tautan, dan banyak elemen lainnya ke halaman web.
- Struktur : Ini membantu mengorganisir halaman dan membuatnya mudah dinavigasi.

PHP (Hypertext Preprocessor) :

- Dinamis : PHP memungkinkan kita untuk membuat halaman web yang dinamis. Artinya, kita bisa menghasilkan konten berdasarkan data atau kondisi tertentu. Misalnya, tampilkan pesan yang berbeda untuk setiap pengunjung.
- Interaksi : PHP memungkinkan kita untuk berinteraksi dengan basis data, mengirim dan menerima data dari formulir, dan melakukan tugas lain yang membuat situs web lebih interaktif.
- Fleksibilitas : PHP adalah salah satu bahasa yang paling umum digunakan untuk pengembangan web. Banyak platform dan CMS (Content Management System) seperti WordPress juga menggunakan PHP sebagai basisnya.

Jadi, saat kita menggabungkan HTML dan PHP, kita dapat menciptakan halaman web yang tidak hanya terlihat indah, tetapi juga memiliki fungsi yang kuat. Contoh sederhana "Hello World" mungkin terlihat sepele, tetapi itu adalah dasar yang penting untuk memahami cara bekerja dengan HTML dan PHP.

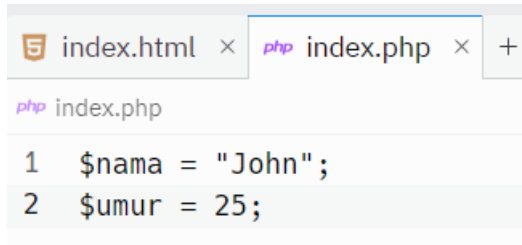
Jangan ragu untuk mulai mencoba dan bereksperimen dengan kode. Fokus utama adalah belajar dan bersenang-senang. Dunia pemrograman web menawarkan banyak peluang kreatif, jadi nikmatilah perjalanannya!

## **Sub-bab 2.2 Variabel dan Tipe Data**

Halo kembali, programmer! Sekarang, kita akan membahas sesuatu yang sangat penting dalam pemrograman: variabel dan tipe data. Jangan khawatir, konsep ini sangat sederhana dan sangat membantu.

**Variabel** adalah seperti wadah tempat kita menyimpan data. Ini seperti kotak yang kita beri nama dan masukkan sesuatu ke dalamnya. Variabel memungkinkan kita untuk menyimpan informasi yang dapat kita gunakan nanti dalam program kita.

Contoh sederhana variabel menggunakan PHP:

A screenshot of a code editor window. The title bar shows two tabs: 'index.html' and 'php index.php'. The active tab is 'php index.php'. The code content is:

```
1 $nama = "John";  
2 $umur = 25;
```

*Gambar 6 Variabel PHP*

Di sini, kita membuat dua variabel. Satu untuk nama dan satu untuk umur. Variabel `$nama` berisi teks "John," dan variabel `$umur` berisi angka 25.

Tipe data juga penting. Mereka memberi tahu komputer bagaimana kita akan memperlakukan data dalam variabel.

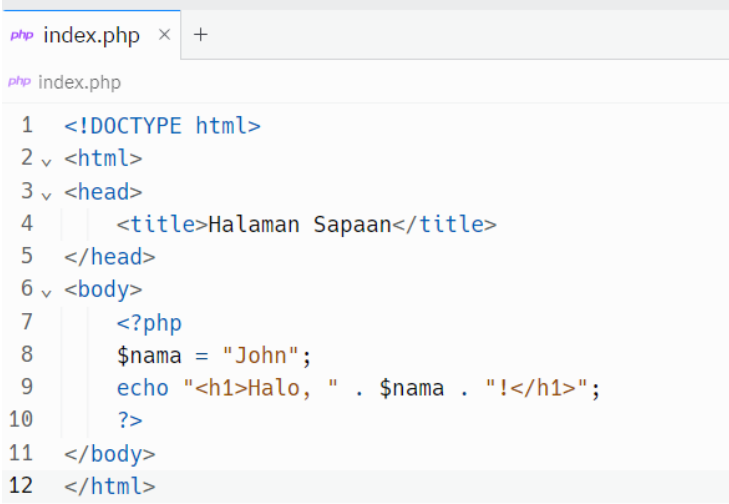
Beberapa tipe data umum dalam PHP adalah:

- **String** : Untuk teks. Contoh: `"Hello, World!"`.
- **Integer** : Untuk angka bulat. Contoh: `42`.
- **Float** : Untuk angka desimal. Contoh: `3.14`.

- **Boolean** : Untuk nilai benar atau salah. Contoh: `true` atau `false`.

**HTML** sendiri tidak memiliki variabel seperti PHP. Namun, HTML dan PHP sering digunakan bersama-sama di halaman web. Variabel PHP dapat digunakan untuk mengubah konten HTML dinamis.

Contoh penggunaan variabel dalam HTML dan PHP:



```
php index.php × +
php index.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Halaman Sapaan</title>
5 </head>
6 <body>
7   <?php
8     $nama = "John";
9     echo "<h1>Halo, " . $nama . "!</h1>";
10    ?>
11 </body>
12 </html>
```

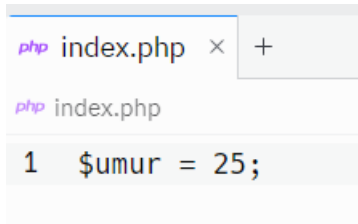
*Gambar 7 Variabel pada HTML dan PHP*

Di sini, kita menggunakan variabel ``$nama`` dalam kode PHP untuk menghasilkan pesan sapaan dalam elemen HTML `<h1>`. Hasilnya adalah "Halo, John!"

Variabel adalah alat yang kuat dalam pemrograman. Mereka memungkinkan kita untuk menyimpan, mengelola, dan memanipulasi data. Dengan pemahaman tentang variabel dan tipe data, kita dapat melakukan banyak hal menarik dalam pemrograman web.

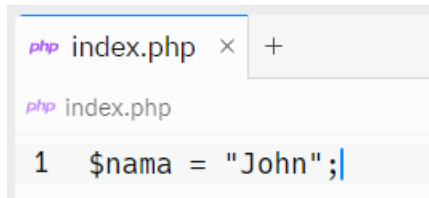
Tipe data adalah panduan yang memberi tahu komputer cara memahami dan memperlakukan data dalam variabel. Jadi, saat kita mendeklarasikan variabel, kita juga memberikan petunjuk tentang tipe data yang akan diisinya.

Misalnya, jika kita ingin menyimpan umur seseorang, kita akan menggunakan tipe data integer, seperti ini:

A screenshot of a code editor window titled 'index.php'. The editor shows a single line of PHP code: `1 $umur = 25;`. The text is highlighted in a light gray background.

```
php index.php × +
php index.php
1 $umur = 25;
```

Tetapi jika kita ingin menyimpan nama, kita akan menggunakan tipe data string, seperti ini:

A screenshot of a code editor window titled 'index.php'. The editor shows a single line of PHP code: `1 $nama = "John";|`. The text is highlighted in a light gray background.

```
php index.php × +
php index.php
1 $nama = "John";|
```

Tipe data memungkinkan kita untuk melakukan operasi yang masuk akal pada variabel. Jika kita mencoba menjumlahkan dua string, ini akan menjadi aneh, tetapi jika kita menjumlahkan dua integer, itu akan berfungsi dengan baik.

Di samping tipe data dasar, seperti string dan integer, PHP juga mendukung tipe data lain, seperti float

untuk angka desimal, boolean untuk nilai benar atau salah, dan array untuk mengelompokkan data.

Contoh penggunaan tipe data lainnya:

```
php index.php × +
php index.php
1 $pi = 3.14; // float
2 $isAdult = true; // boolean
3 $daftarBelanjaan = array("apel", "pisang", "jeruk"); // array
4
```

Variabel dan tipe data adalah bagian penting dalam pemrograman. Mereka memungkinkan kita untuk mengoperasikan dan memanipulasi data dengan cara yang membuat program kita lebih pintar dan berguna.

Sekarang, giliranmu untuk mencoba! Buat beberapa variabel, isi dengan berbagai tipe data, dan lihat apa yang bisa kamu lakukan dengan mereka. Selamat bersenang-senang dan selamat belajar, programmer!



## Sub-bab 2.3 Operator dan Ekspresi

Hai kawan programmer! Kita telah belajar tentang variabel dan tipe data, sekarang saatnya untuk memahami bagaimana kita dapat melakukan operasi dan perhitungan dengan data ini menggunakan operator dan ekspresi.

**Operator** adalah seperti simbol yang memberi tahu komputer apa yang harus dilakukan dengan data. Ada beberapa operator yang berbeda dalam PHP. Mari kita lihat beberapa yang paling umum:

1. **Operator Aritmatika**, Digunakan untuk perhitungan matematika.

- `+` (penambahan)

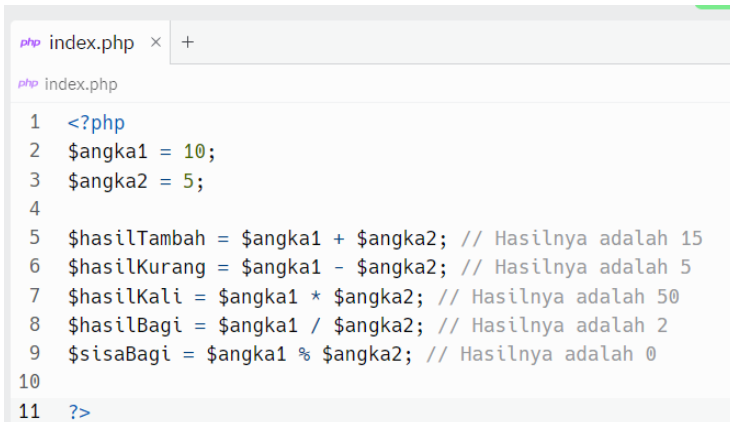
- `-` (pengurangan)

- `\*` (perkalian)

- `/` (pembagian)

- `%` (modulo - sisa hasil bagi)

Contoh penggunaan operator aritmatika:



```
php index.php × +
php index.php
1 <?php
2 $angka1 = 10;
3 $angka2 = 5;
4
5 $hasilTambah = $angka1 + $angka2; // Hasilnya adalah 15
6 $hasilKurang = $angka1 - $angka2; // Hasilnya adalah 5
7 $hasilKali = $angka1 * $angka2; // Hasilnya adalah 50
8 $hasilBagi = $angka1 / $angka2; // Hasilnya adalah 2
9 $sisaBagi = $angka1 % $angka2; // Hasilnya adalah 0
10
11 ?>
```

Gambar 8 Penggunaan Operator Aritmatika

2. **Operator Perbandingan**, Digunakan untuk membandingkan dua nilai.

- `==` (sama dengan)

- `!=` (tidak sama dengan)

- `<` (kurang dari)

- `>` (lebih dari)

- `<=` (kurang dari atau sama dengan)
- `>=` (lebih dari atau sama dengan)

Contoh penggunaan operator perbandingan:

```

php index.php × +
php index.php
1 <?php
2 $nilai1 = 10;
3 $nilai2 = 5;
4
5 $samaDengan = $nilai1 == $nilai2; // Hasilnya adalah false
6 $tidakSamaDengan = $nilai1 != $nilai2; // Hasilnya adalah true
7 $lebihDari = $nilai1 > $nilai2; // Hasilnya adalah true
8
9
10 ?>
Generate

```

*Gambar 9 Operator Perbandingan*

**3. Operator Logika**, Digunakan untuk menggabungkan pernyataan logika.

- `&&` (dan)
- `||` (atau)
- `!` (bukan)

Contoh penggunaan operator logika:



```
php index.php × +
php index.php
1 <?php
2 $benar = true;
3 $salah = false;
4
5 $hasilDan = $benar && $salah; // Hasilnya adalah false
6 $hasilAtau = $benar || $salah; // Hasilnya adalah true
7 $hasilBukan = !$benar; // Hasilnya adalah false
8 ?>
```

*Gambar 10 Operator Logika*

**Ekspresi** adalah kombinasi variabel dan operator yang menghasilkan nilai. Ini adalah cara kita melakukan perhitungan dan pengambilan keputusan dalam program.

Contoh ekspresi:

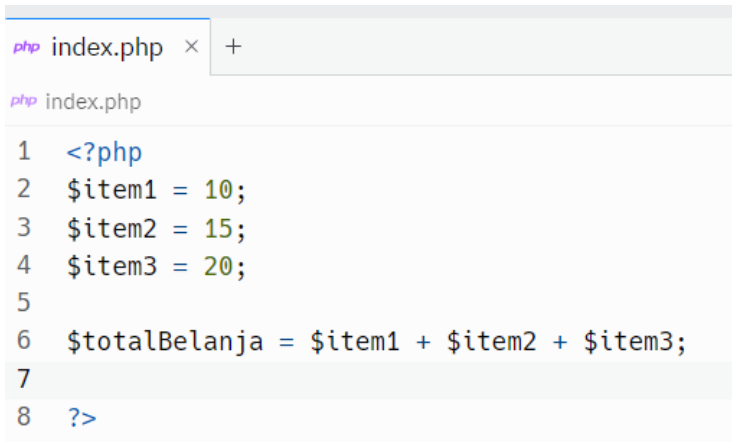
```
php index.php × +
php index.php
1 <?php
2 $panjang = 10;
3 $lebar = 5;
4 $luas = $panjang * $lebar; // Hasilnya adalah 50
5 echo "Luas persegi panjang adalah $luas";
6 ?>
```

*Gambar 11 Ekspresi*

Dengan operator dan ekspresi, kita dapat membuat program yang lebih pintar dan dinamis. Mereka membantu kita mengambil keputusan, melakukan perhitungan, dan banyak lagi.

Saat kita memahami cara operator dan ekspresi berfungsi, kita bisa melihat betapa berguna mereka dalam pemrograman.

Misalkan kita ingin membuat program yang menghitung total belanjaan dari beberapa item:



```
php index.php × +
php index.php
1 <?php
2 $item1 = 10;
3 $item2 = 15;
4 $item3 = 20;
5
6 $totalBelanja = $item1 + $item2 + $item3;
7
8 ?>
```

*Gambar 12 Beberapa item*

Dalam contoh ini, kita menggunakan operator penambahan (^+`) untuk menghitung total belanjaan. Dengan ekspresi ini, kita bisa dengan mudah mengubah jumlah item yang dihitung tanpa harus menulis ulang seluruh kode.

Selain itu, operator perbandingan digunakan untuk pengambilan keputusan. Misalnya:



```
php index.php × +
php index.php
1 <?php
2 $umur = 25;
3
4 if ($umur >= 18) {
5     echo "Kamu adalah seorang dewasa.";
6 } else {
7     echo "Kamu masih seorang anak.";
8 }
9
10 ?>
```

*Gambar 13 Operator Perbandingan*

Di sini, kita menggunakan operator perbandingan (`>=`) untuk memeriksa jika seseorang berusia 18 tahun atau lebih. Berdasarkan hasilnya, kita mencetak pesan yang sesuai.

Operator dan ekspresi adalah alat yang sangat berguna dalam pemrograman. Mereka memungkinkan kita untuk melakukan perhitungan, pengambilan keputusan, dan banyak operasi lainnya dalam program kita.

Sekarang, cobalah bermain-main dengan operator dan ekspresi. Lakukan beberapa perhitungan sederhana dan lihat apa yang bisa kamu capai. Selamat bersenang-senang dan selamat belajar, teman-teman programmer!

## **Sub-bab 2.4 Struktur Kontrol Percabangan**

Hei teman-teman pemrogram muda! Sekarang, kita akan memasuki dunia yang sangat menarik dalam pemrograman: **struktur kontrol percabangan**. Ini seperti bercerita pada komputer apa yang harus dilakukan dalam berbagai situasi. Dengan kata lain, kita memberikan instruksi tentang apa yang harus dilakukan dalam skenario tertentu.

Ada dua jenis percabangan yang umum digunakan dalam pemrograman: **if** dan **else**. Mari kita bahas keduanya.



**If Statement** adalah cara komputer membuat keputusan. Ini berfungsi dengan cara berikut:

```
php index.php × +
php index.php
1 <?php
2 if (kondisi) {
3     // kode yang dijalankan jika kondisi benar
4 }
5 ?>
```

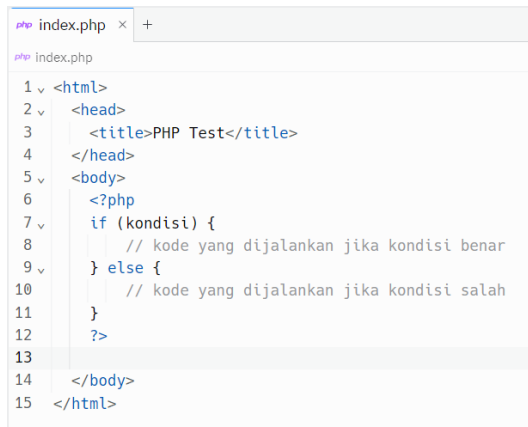
Contoh penggunaan If Statement:

```
php index.php × +
php index.php
1 <?php
2 $umur = 18;
3
4 if ($umur >= 18) {
5     echo "Kamu adalah seorang dewasa.";
6 }
7 ?>
8 ?>
```

*Gambar 14 Penggunaan If Statement*

Pada contoh di atas, jika umur sama dengan atau lebih besar dari 18, pesan "Kamu adalah seorang dewasa" akan dicetak.

**Else Statement** adalah cara komputer membuat keputusan alternatif:



```
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     if (kondisi) {
8       // kode yang dijalankan jika kondisi benar
9     } else {
10      // kode yang dijalankan jika kondisi salah
11    }
12    ?>
13
14 </body>
15 </html>
```

*Gambar 15 Penggunaan Else Statement*

Contoh penggunaan Else Statement:

```
php index.php x +
php index.php
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     $umur = 15;
8
9     if ($umur >= 18) {
10      echo "Kamu adalah seorang dewasa.";
11    } else {
12      echo "Kamu masih seorang anak.";
13    }
14   ?>
15
16 </body>
17 </html>
```

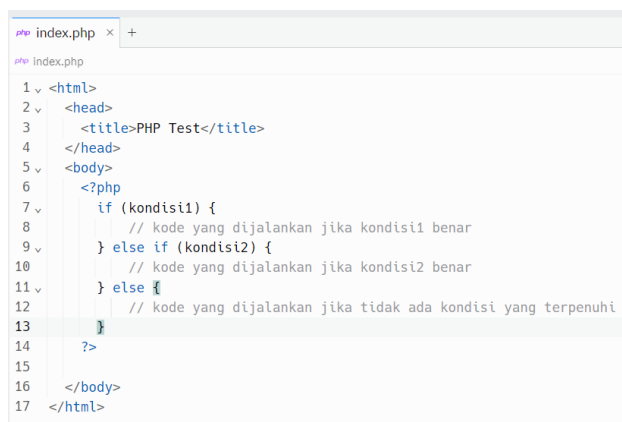
*Gambar 16 Penggunaan Else Statement*

Pada contoh di atas, jika umur kurang dari 18, pesan "Kamu masih seorang anak" akan dicetak.

Struktur kontrol percabangan ini sangat bermanfaat ketika kita ingin mengubah perilaku program berdasarkan situasi. Contohnya, dalam game, kita dapat menggunakan percabangan untuk memutuskan apa yang harus terjadi jika pemain memenangkan permainan atau kehilangan permainan.

Cobalah untuk bermain-main dengan pernyataan If dan Else. Kamu dapat membuat program yang lebih pintar dengan penggunaan yang kreatif. Selamat bersenang-senang dan selamat belajar, programmer pemula!


Selain If dan Else, kita juga memiliki **Else If**. Ini adalah cara komputer membuat lebih banyak pilihan. Jika kondisi pertama tidak terpenuhi, maka komputer akan memeriksa kondisi kedua, dan seterusnya.



```
php index.php x +
php index.php
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     if (kondisi1) {
8       // kode yang dijalankan jika kondisi1 benar
9     } else if (kondisi2) {
10      // kode yang dijalankan jika kondisi2 benar
11    } else {
12      // kode yang dijalankan jika tidak ada kondisi yang terpenuhi
13    }
14  ?>
15
16 </body>
17 </html>
```

Gambar 17 Penggunaan Else If

## Contoh penggunaan Else If:

A screenshot of a code editor window titled 'index.php'. The code is written in PHP and HTML. It defines a variable \$nilai = 85; and uses an if-else-if-else structure to check the value of \$nilai. The conditions are: if (\$nilai >= 90) { echo "Kamu mendapat A."; }, else if (\$nilai >= 80) { echo "Kamu mendapat B."; }, else if (\$nilai >= 70) { echo "Kamu mendapat C."; }, else { echo "Kamu perlu belajar lebih giat!"; }. The code is enclosed in HTML tags: <html>, <head>, <title>PHP Test</title>, </head>, <body>, <?php, and </body>, </html>. The line numbers 1 through 21 are visible on the left side of the editor.

```
1 <html>
2 <head>
3 <title>PHP Test</title>
4 </head>
5 <body>
6 <?php
7     $nilai = 85;
8
9     if ($nilai >= 90) {
10         echo "Kamu mendapat A.";
11     } else if ($nilai >= 80) {
12         echo "Kamu mendapat B.";
13     } else if ($nilai >= 70) {
14         echo "Kamu mendapat C.";
15     } else {
16         echo "Kamu perlu belajar lebih giat!";
17     }
18     ?>
19
20 </body>
21 </html>
```

*Gambar 18 Penggunaan Else If*

Pada contoh di atas, komputer memeriksa beberapa kondisi berturut-turut dan mencetak pesan yang sesuai dengan kondisi pertama yang terpenuhi.

Struktur kontrol percabangan ini memungkinkan kita membuat program yang cerdas dan responsif terhadap situasi yang berbeda. Mereka sangat penting

dalam memprogram game, aplikasi, dan banyak hal lainnya.

Selanjutnya, kita akan mempelajari lebih lanjut tentang **looping** yang akan memungkinkan kita melakukan tugas yang sama berulang kali. Teruslah berlatih dan jangan ragu untuk mencoba-coba dengan percabangan ini. Selamat belajar dan semoga menjadi programmer hkamul!

## **Sub-bab 2.5 Struktur Kontrol Perulangan**

Hai teman-teman pemrogram muda! Sekarang, kita akan membahas sesuatu yang sangat seru dalam pemrograman, yaitu **struktur kontrol perulangan**. Ini adalah cara komputer menjalankan tugas-tugas berulang tanpa harus menulis kode berkali-kali.

Ada dua jenis perulangan yang umum digunakan dalam pemrograman: **for loop** dan **while loop**. Mari kita bahas keduanya.

**For Loop** adalah cara yang bagus untuk menjalankan perintah berulang kali berdasarkan jumlah iterasi tertentu. Ini berfungsi dengan cara berikut:

```
<?php
    for ($i = 0; $i < 5; $i++) {
        // kode yang diulang 5 kali
    }
?>
```

Contoh penggunaan For Loop:

```
php index.php x +
php index.php
1 <html>
2 <head>
3 <title>PHP Test</title>
4 </head>
5 <body>
6 <?php
7 for ($i = 0; $i < 5; $i++) {
8     echo "Iterasi ke- $i$ <br>";
9 }
10 ?>
11
12 </body>
13 </html>
```

Gambar 19 Penggunaan For Loop

Hasil dari contoh di atas adalah mencetak "Iterasi ke-0" hingga "Iterasi ke-4". Loop ini berjalan 5 kali, sesuai dengan kondisi yang diberikan.

**While Loop** adalah cara untuk menjalankan perintah berulang kali selama kondisi tertentu terpenuhi. Ini berfungsi seperti ini:



```
<?php
while (kondisi) {
    // kode yang diulang selama kondisi terpenuhi
}
?>
```

Contoh penggunaan While Loop:

A screenshot of a code editor window titled 'index.php'. The code is as follows:

```
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     $angka = 1;
8
9     while ($angka <= 5) {
10      echo "Angka: $angka<br>";
11      $angka++;
12    }
13   ?>
14
15 </body>
16 </html>
```

Gambar 20 Penggunaan While Loop

Loop ini akan berjalan selama ` \$angka ` kurang dari atau sama dengan 5. Hasilnya adalah mencetak "Angka: 1" hingga "Angka: 5".

Struktur kontrol perulangan ini sangat bermanfaat ketika kita perlu melakukan tugas-tugas yang berulang, seperti menghitung total harga belanjaan, mencetak deret bilangan, atau melakukan operasi lain berulang kali.

Selain **For Loop** dan **While Loop**, kita juga memiliki **Do-While Loop**. Ini adalah cara komputer menjalankan perintah setidaknya satu kali dan kemudian terus menjalankannya selama kondisi tertentu terpenuhi.

```
<?php
do {
    // kode yang dijalankan minimal satu kali
} while (kondisi);
?> |
```

Contoh penggunaan Do-While Loop:

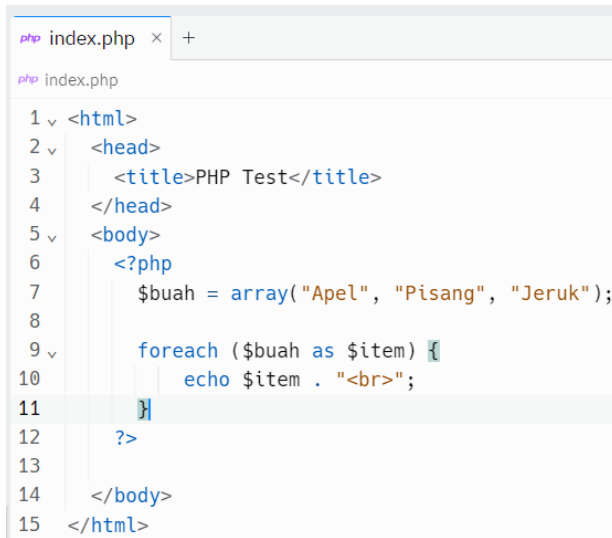
```
php index.php x +
php index.php
1 <html>
2 <head>
3 <title>PHP Test</title>
4 </head>
5 <body>
6 <?php
7     $angka = 1;
8
9     do {
10        echo "Angka: $angka<br>";
11        $angka++;
12    } while ($angka <= 5);
13    ?>
14
15 </body>
16 </html>
```

Gambar 21 Penggunaan Do While

Loop ini akan selalu menjalankan kode setidaknya sekali, bahkan jika kondisi awalnya tidak terpenuhi. Kemudian, ia akan terus berjalan selama ` \$angka ` kurang dari atau sama dengan 5.

Selain itu, dalam pemrograman, kita sering menghadapi situasi di mana kita perlu mengulang daftar item atau elemen dalam sebuah array. Kita dapat

menggunakan perulangan **foreach** untuk melakukan ini dengan mudah.



```
php index.php x +
php index.php
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     $buah = array("Apel", "Pisang", "Jeruk");
8
9     foreach ($buah as $item) {
10      echo $item . "<br>";
11    }
12  ?>
13
14 </body>
15 </html>
```

*Gambar 22 Penggunaan Foreach*

Loop ini akan mencetak setiap item dalam array buah.

Perulangan adalah alat yang sangat berguna dalam pemrograman. Mereka memungkinkan kita untuk melakukan tugas-tugas berulang dengan efisien. Praktikkan dan eksplorasi berbagai jenis loop ini, dan Kamu akan menjadi programmer yang semakin hebat!

Selamat belajar, dan jangan lupa selalu bermain-main dengan kode. Kreativitas adalah kunci dalam pemrograman.

## BAB 3

### STRUKTUR DATA DASAR

#### Sub-bab 3.1 Array dan List

Hei, teman-teman pemrogram! Sekarang kita akan membahas konsep yang sangat keren dalam pemrograman, yaitu **array**. Bayangkan array seperti sebuah kotak besar di mana Kamu dapat menyimpan banyak hal. Setiap item dalam kotak memiliki nomor yang disebut **indeks** untuk mengidentifikasinya.

Misalnya, mari kita coba membuat sebuah array yang menyimpan beberapa buah:

```
<?php
$buah = array("Apel", "Pisang", "Jeruk");
?>
```

*Gambar 23 Contoh Array untuk Buah*

Dalam contoh ini, `\$buah` adalah array, dan setiap item (Apel, Pisang, Jeruk) memiliki indeks. Indeks dimulai dari 0, jadi Apel memiliki indeks 0, Pisang indeks 1, dan Jeruk indeks 2.

Kamu dapat mengakses item dalam array dengan cara ini:


```
echo $buah[0]; // Akan mencetak "Apel"  
echo $buah[1]; // Akan mencetak "Pisang"  
echo $buah[2]; // Akan mencetak "Jeruk"
```

*Gambar 24 Mengakses Array untuk Buah*

Tentu saja, array bisa lebih besar dan berisi lebih banyak jenis data. Ini membantu kita menyusun dan mengelola informasi dengan lebih baik.

**List** adalah cara lain untuk mengelola kumpulan data. Dalam PHP, kita sering menggunakan array untuk tujuan yang serupa, tetapi dalam beberapa bahasa pemrograman lain, "list" adalah struktur data khusus. Misalnya, kita bisa menggunakan list untuk menyimpan data seperti nama, alamat email, dan nomor telepon pelanggan.

Mari kita bermain dengan array dan list:

A screenshot of a code editor window titled 'index.php'. The code is written in PHP and is enclosed in HTML tags. It demonstrates array operations: creating an array of fruits, adding an item, removing an item, counting items, and displaying them. The code is as follows:

```
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6   <?php
7     $buah = array("Apel", "Pisang", "Jeruk");
8
9     // Menambahkan item ke array
10    $buah[] = "Anggur";
11
12    // Menghapus item dari array
13    unset($buah[1]);
14
15    // Menghitung jumlah item dalam array
16    $jumlahBuah = count($buah);
17
18    // Menampilkan semua item dalam array
19    for ($i = 0; $i < $jumlahBuah; $i++) {
20      echo "Buah: " . $buah[$i] . "<br>";
21    }
22  ?>
23
24 </body>
25 </html>
```

Gambar 25 Array dan List

Ini hanya permulaan, teman-teman. Array dan list adalah alat yang sangat berguna dalam pemrograman.

Kamu dapat menggunakannya untuk menyimpan dan mengelola banyak data dengan cara yang efisien.

Selain membuat array secara manual seperti yang kita lakukan sebelumnya, Kamu juga dapat membuat array yang lebih kompleks dan dinamis. Misalnya, array asosiatif memungkinkan Kamu untuk menggunakan kunci bukan hanya indeks numerik untuk mengidentifikasi elemen. Kunci ini bisa berupa string yang menjelaskan elemen tersebut.

```
<?php
// Membuat array asosiatif
$siswa = array("nama" => "Anita", "usia" => 15, "kelas" => "XII-A");

// Mengakses elemen dengan kunci
echo "Nama: " . $siswa["nama"] . "<br>";
echo "Usia: " . $siswa["usia"] . " tahun<br>";
echo "Kelas: " . $siswa["kelas"];
?>
```

*Gambar 26 Array Asosiatif*

Pada contoh di atas, kita membuat array asosiatif yang berisi informasi tentang seorang siswa. Kunci (nama, usia, kelas) digunakan untuk mengakses nilai yang sesuai.



Kamu juga dapat menggunakan perulangan untuk mengiterasi melalui elemen-elemen array, baik array numerik maupun asosiatif. Ini memungkinkan Kamu untuk mengolah data dengan lebih efisien, terutama ketika Kamu memiliki banyak elemen.

```
<?php
    $buah = array("Apel", "Pisang", "Jeruk", "Anggur");

    // Menggunakan perulangan untuk mencetak semua buah
    foreach ($buah as $item) {
        echo "Buah: " . $item . "<br>";
    }
?>
```

Perulangan `foreach` di atas akan mengambil setiap elemen dalam array `\$buah` dan mencetaknya. Dengan demikian, Kamu dapat mengelola daftar item dengan mudah, terlepas dari seberapa banyaknya.

Array dan list adalah salah satu konsep paling penting dalam pemrograman. Mereka memungkinkan Kamu untuk mengatur, menyimpan, dan mengakses data dengan efisien. Jadi, jangan ragu untuk bermain-main dengan array, dan ciptakan kode yang hebat!

Selamat belajar, dan ingatlah bahwa pemrograman adalah seperti memecahkan teka-teki yang menyenangkan.

### **Sub-bab 3.2 String dan Teks Processing**

Hai teman-teman! Sekarang kita akan membahas sesuatu yang sangat penting dalam pemrograman, yaitu **string**. String adalah kumpulan karakter, seperti huruf, angka, dan simbol yang digabungkan bersama.

Pertama, mari kita coba membuat sebuah string sederhana:

```
$nama = "John";  
$pesan = "Halo, $nama!";
```

*Gambar 27 String*

Dalam contoh di atas, kita memiliki dua string: ``$nama`` dan ``$pesan``. String dapat berisi teks apa pun yang Kamu inginkan.

Kamu dapat menggabungkan string dengan operator titik (`.`):

```
$greeting = "Halo, " . $nama . "!";|
```

*Gambar 28 Menggabungkan String*

Selain itu, PHP memiliki banyak fungsi yang dapat Kamu gunakan untuk memanipulasi string. Misalnya, untuk menghitung panjang sebuah string, Kamu dapat menggunakan `strlen()`:

```
<?php
    $teks = "Ini adalah contoh teks.";
    $panjang = strlen($teks);
    echo "Panjang teks: $panjang karakter";|
?>
```

*Gambar 29 Menghitung Panjang String*

String juga dapat dibagi menjadi potongan-potongan kecil dengan menggunakan fungsi `substr()`:

```
<?php
    $teks = "Ini adalah contoh teks.";
    $potongan = substr($teks, 0, 3);
    echo "Potongan pertama: $potongan";|
?>
```

*Gambar 30 Membagi String*

Selain itu, Kamu dapat mencari dan mengganti teks dalam string dengan fungsi `str_replace()`:

```
<?php
    $teks = "Saya suka makan nasi.";
    $teksBaru = str_replace("nasi", "pizza", $teks);
    echo $teksBaru;
?>
```

*Gambar 31 Mengganti Teks dalam String*

Ini hanya permulaan, teman-teman! String adalah salah satu elemen dasar dalam pemrograman, dan Kamu akan menggunakannya dalam hampir setiap program yang Kamu tulis.

Sekarang, mari jelajahi beberapa operasi lain yang bisa kita lakukan dengan string:

**1. Penggabungan String (Concatenation)**, Kita sudah melihat penggabungan string dengan operator titik (``.``). Ini memungkinkan kita untuk menggabungkan beberapa string menjadi satu.

```

<?php
    $namaDepan = "John";
    $namaBelakang = "Doe";
    $namaLengkap = $namaDepan . " " . $namaBelakang;
    echo "Nama lengkap: $namaLengkap";
?>

```

*Gambar 32 Penggabungan String*

2. **Pencarian Teks dalam String**, Kamu dapat menggunakan fungsi `strpos()` untuk menemukan posisi pertama suatu teks dalam string.

```

<?php
    $kalimat = "Halo, nama saya John.";
    $posisi = strpos($kalimat, "John");
    echo "Kata 'John' ditemukan pada posisi ke-{$posisi}";
?>

```

*Gambar 33 Pencarian Teks String*

3. **Pemotongan String (Substring)**, Dengan fungsi `substr()`, Kamu dapat memotong sebagian teks dari string.

```

<?php
    $teks = "Ini adalah contoh teks.";
    $potongan = substr($teks, 8, 7); // Mulai dari indeks 8, sepanjang 7
    karakter
    echo "Potongan teks: $potongan";
?>

```

*Gambar 34 Pemotongan String*

4. **Penggantian Teks dalam String**, Fungsi `str_replace()` memungkinkan Kamu untuk mengganti

semua kemunculan teks tertentu dengan teks lain dalam sebuah string.

```
<?php
$teks = "Saya suka makan nasi.";
$teksBaru = str_replace("nasi", "pizza", $teks);
echo $teksBaru;
?>
```

*Gambar 35 Mengganti Teks dalam String*

5. **Konversi Huruf**, Kamu dapat mengonversi huruf dalam string menjadi huruf kapital atau huruf kecil dengan fungsi `strtoupper()` dan `strtolower()`.

```
<?php
$teks = "Ini Adalah TEKS Contoh.";
$hurufKecil = strtolower($teks);
$hurufBesar = strtoupper($teks);
echo "Huruf kecil: $hurufKecil<br>";
echo "Huruf besar: $hurufBesar";
?>
```

*Gambar 36 Konversi dalam String*

String adalah bagian yang sangat penting dalam pemrograman karena sebagian besar data yang kita proses adalah teks. Semakin Kamu berlatih, semakin paham Kamu dalam memanipulasi string dan data teks. Selamat belajar, dan jangan ragu untuk bereksperimen dengan operasi-operasi string ini!

## Sub-bab 3.3 Tabel dan Matriks

Halo kembali, teman-teman! Kali ini kita akan membahas konsep yang seru, yaitu **tabel** dan **matriks**. Dalam pemrograman, tabel dan matriks adalah cara hebat untuk mengatur data dalam bentuk yang rapi dan terstruktur.

### Tabel

Dalam HTML, tabel dibuat dengan menggunakan elemen `<table>`, dan kita dapat mengisi sel-selnya dengan data. Mari lihat contohnya:

```

1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6 <table>
7 <tr>
8   <th>Nama</th>
9   <th>Usia</th>
10 </tr>
11 <tr>
12   <td>John</td>
13   <td>25</td>
14 </tr>
15 <tr>
16   <td>Anita</td>
17   <td>20</td>
18 </tr>
19 </table>
20 </body>
21 </html>

```

*Gambar 37 Tabel HTML*

Dalam contoh di atas, kita memiliki tabel sederhana yang berisi nama dan usia. Tabel ini terdiri dari baris-baris (elemen ``<tr>``) dan sel-sel (elemen ``<td>``). Elemen ``<th>`` digunakan untuk judul kolom. Kamu dapat membuat tabel lebih besar dengan menambahkan lebih banyak baris dan kolom.



## Matriks

Dalam pemrograman, kita sering menggunakan matriks untuk menyimpan data dalam bentuk dua dimensi. Ini seperti tabel di mana Kamu memiliki baris dan kolom. Dalam PHP, Kamu dapat membuat matriks sebagai berikut:

```
<?php
    $matriks = array(
        array(1, 2, 3),
        array(4, 5, 6),
        array(7, 8, 9)
    );
?>
```

*Gambar 38 Matrix*

Dalam contoh di atas, kita memiliki matriks 3x3 yang berisi angka-angka. Kamu dapat mengakses elemen matriks dengan menggunakan indeks baris dan kolom:

```
echo $matriks[1][2]; // Akan mencetak 6
```

Kamu juga dapat menggunakan perulangan untuk mengiterasi melalui matriks:

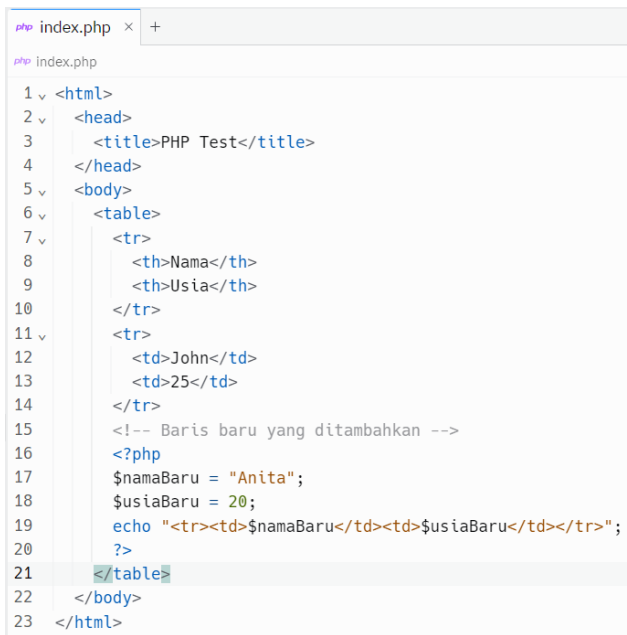
```
<?php
for ($i = 0; $i < 3; $i++) {
    for ($j = 0; $j < 3; $j++) {
        echo $matriks[$i][$j] . " ";
    }
    echo "<br>";
}
?>
```

Dengan tabel dan matriks, Kamu dapat mengatur dan memanipulasi data dengan cara yang kuat dan terstruktur. Ini sangat berguna saat Kamu bekerja dengan data yang lebih kompleks dalam pemrograman.

Ketika Kamu bekerja dengan tabel dan matriks, seringkali Kamu perlu melakukan beberapa operasi, seperti menambahkan atau menghapus data. Berikut adalah beberapa hal yang dapat Kamu lakukan:

## Menambah Baris ke Tabel

Kamu dapat menambahkan baris baru ke tabel HTML dengan menggunakan JavaScript atau PHP. Mari kita lihat contoh sederhana dengan PHP:



```
1 <html>
2 <head>
3   <title>PHP Test</title>
4 </head>
5 <body>
6 <table>
7   <tr>
8     <th>Nama</th>
9     <th>Usia</th>
10  </tr>
11 <tr>
12   <td>John</td>
13   <td>25</td>
14 </tr>
15   <!-- Baris baru yang ditambahkan -->
16   <?php
17     $namaBaru = "Anita";
18     $usiaBaru = 20;
19     echo "<tr><td>$namaBaru</td><td>$usiaBaru</td></tr>";
20   ?>
21 </table>
22 </body>
23 </html>
```

Gambar 39 Menambahkan Baris ke Tabel HTML

Kamu juga dapat menggunakan JavaScript untuk menambahkan baris secara dinamis saat pengguna berinteraksi dengan halaman web.

## Operasi dengan Matriks

Dalam pemrograman, Kamu akan sering memanipulasi data dalam matriks. Kamu dapat menambahkan, menghapus, atau mengubah elemen-elemen matriks sesuai kebutuhan.

Contoh sederhana, menambahkan baris baru ke matriks PHP:

```
<?php
    $matriks = array(
        array(1, 2, 3),
        array(4, 5, 6)
    );

    $barisBaru = array(7, 8, 9);
    array_push($matriks, $barisBaru);|
?>
```

*Gambar 40 Menambahkan Baris Baru ke Matrik*

Ini akan menambahkan baris baru ke matriks. Kamu juga dapat menggunakan ``array_pop()`` untuk menghapus elemen dari baris terakhir atau fungsi lain untuk melakukan berbagai operasi dengan matriks.

Ketika Kamu mulai bekerja dengan tabel dan matriks dalam pemrograman, Kamu akan menemukan bahwa mereka adalah alat yang sangat berguna untuk menyimpan dan mengelola data. Mereka memungkinkan Kamu untuk mengorganisir informasi dengan cara yang lebih efisien.

Jadi, lanjutkan belajar dan bereksperimen dengan tabel dan matriks, dan jangan ragu untuk menjadikannya bagian dari proyek-proyek pemrograman Kamu!

### **Sub-bab 3.4 Penggunaan Data Structures**

Hai teman-teman! Sekarang, kita akan membahas sesuatu yang sangat penting dalam pemrograman, yaitu **struktur data**. Struktur data adalah cara untuk mengatur dan menyimpan data dengan cara yang efisien.

#### **Array Asosiatif**

Array asosiatif adalah jenis struktur data yang memungkinkan kita untuk menyimpan data dengan kunci unik. Ini seperti kamus di mana setiap kata memiliki

definisi. Dalam PHP, kita dapat membuat array asosiatif sebagai berikut:

```
<?php
    $mahasiswa = array(
        "nama" => "John",
        "usia" => 25,
        "jurusan" => "Informatika"
    );
?>
```

*Gambar 41 Membuat Array Assosiatif dalam PHP*

Kamu dapat mengakses nilai dengan menggunakan kunci:

```
echo "Nama: " . $mahasiswa["nama"];
echo "Usia: " . $mahasiswa["usia"];
```

## Objek

Objek adalah struktur data yang lebih kompleks. Mereka menggabungkan data dan metode (fungsi) yang beroperasi pada data tersebut. Mari lihat contoh sederhana:

```

<?php
class Mahasiswa {
    public $nama;
    public $usia;

    function sapa() {
        echo "Halo, saya $this->nama!";
    }
}

$mahasiswa1 = new Mahasiswa();
$mahasiswa1->nama = "John";
$mahasiswa1->usia = 25;
$mahasiswa1->sapa();|
?>

```

*Gambar 42 Objek*

Dalam contoh di atas, kita membuat sebuah objek `Mahasiswa` dengan properti `nama` dan `usia`, serta metode `sapa()`. Objek ini memungkinkan kita untuk menggabungkan data dan perilaku menjadi satu kesatuan.

## JSON

JSON (JavaScript Object Notation) adalah format umum untuk pertukaran data. Ini mirip dengan array asosiatif dalam PHP. Mari lihat contoh:

```
{
  "nama": "John",
  "usia": 25,
  "jurusan": "Informatika"
}
```

JSON digunakan secara luas dalam pertukaran data antara aplikasi web dan server. PHP memiliki fungsi untuk mengkodekan dan mendekode JSON.

Struktur data adalah komponen penting dalam pemrograman. Mereka memungkinkan Kamu untuk mengorganisir dan mengakses data dengan lebih efisien. Saat Kamu memahami berbagai jenis struktur data, Kamu dapat memilih yang paling cocok untuk setiap tugas.

Saat Kamu bekerja dengan struktur data, ada beberapa operasi umum yang sering Kamu perlukan untuk dilakukan:



## Menambahkan Data

Kamu dapat menambahkan data ke dalam struktur data dengan cara yang sederhana. Misalnya, jika Kamu memiliki array asosiatif seperti ini:

```
$buah = array(  
    "nama" => "Apel",  
    "warna" => "Merah"  
);
```

Kamu dapat dengan mudah menambahkan properti baru:

```
$buah["rasa"] = "Manis";
```

Sama halnya dengan objek:

```
$buah = new stdClass();  
$buah->nama = "Apel";  
$buah->warna = "Merah";  
$buah->rasa = "Manis";
```

## Menghapus Data

Kamu juga dapat menghapus data dari struktur data. Misalnya, untuk menghapus properti dari array asosiatif:

```
unset($buah["warna"]);
```

Untuk menghapus properti dari objek:

```
unset($buah->rasa);
```

## Iterasi Melalui Data

Untuk mengakses semua data dalam struktur data, Kamu dapat menggunakan perulangan. Contohnya, dengan array asosiatif:

```
foreach ($buah as $kunci => $nilai) {  
    echo "$kunci: $nilai<br>";  
}
```

Untuk objek:

```
foreach ($buah as $kunci => $nilai) {  
    echo "$kunci: $nilai<br>";  
}
```

## Serialisasi dan Deserialisasi

Kadang-kadang, Kamu perlu menyimpan struktur data ke dalam format yang dapat disimpan, seperti string. Ini disebut serialisasi. Kamu kemudian dapat

mengembalikannya ke bentuk semula, disebut deserialisasi.

```
$buah = array("Apel", "Jeruk", "Mangga");  
$buah_string = serialize($buah);  
$buah_kembali = unserialize($buah_string);
```

Dengan pemahaman tentang operasi-operasi ini, Kamu dapat lebih fleksibel dalam mengelola data dalam proyek pemrograman Kamu.

Teruslah belajar tentang berbagai struktur data dan bagaimana Kamu dapat menggunakannya dalam pemrograman. Dengan pemahaman yang baik tentang struktur data, Kamu dapat membangun aplikasi yang lebih kuat dan efisien!

## **BAB 4**

### **FUNGSI DAN PROSEDUR**

#### **Sub-bab 4.1 Apa itu Fungsi?**

Hai teman-teman! Sekarang kita akan memasuki dunia yang menarik tentang "fungsi." Fungsi adalah salah

satu konsep paling penting dalam pemrograman. Mari kita bahas apa itu fungsi dan mengapa mereka begitu penting.

## Apa Itu Fungsi?

Pikirkan fungsi seperti resep dalam memasak. Ketika Kamu ingin membuat sesuatu yang spesifik, Kamu mengikuti langkah-langkah tertentu. Fungsi adalah kumpulan langkah-langkah atau perintah yang diberi nama dan dapat digunakan kembali.

Contoh sederhana dalam PHP:

```
<?php
function sapa() {
    echo "Halo, dunia!";
}
?>
```

*Gambar 43 Fungsi*

Fungsi di atas bernama "sapa" dan ketika dipanggil, ia akan mencetak "Halo, dunia!" di layar.

## Mengapa Fungsi Penting?

Fungsi memiliki beberapa manfaat:

1. **Modularitas**, Fungsi memungkinkan Kamu untuk memecah program Kamu menjadi bagian-bagian yang lebih kecil. Ini membuat kode lebih mudah diorganisir dan dimengerti.
2. **Penggunaan Kembali**, Kamu dapat menggunakan fungsi yang sama berkali-kali dalam program Kamu tanpa perlu menulis ulang kode. Ini menghemat waktu dan usaha.
3. **Kesalahan**, Dengan fungsi, jika ada kesalahan dalam program Kamu, Kamu hanya perlu memperbaiki kesalahan di satu tempat (fungsi) daripada mencari di seluruh program.

## Cara Menggunakan Fungsi

Untuk menggunakan fungsi, Kamu harus memanggilnya. Di sini contoh pemanggilan fungsi "sapa":

```
sapa(); // Ini akan mencetak "Halo, dunia!"
```

Kamu juga dapat memberikan argumen ke fungsi. Misalnya, fungsi sapa dengan argumen nama:

```
<?php
function sapa($nama) {
    echo "Halo, $nama!";
}

sapa("John"); // Ini akan mencetak "Halo, John!"
?>
```

## Fungsi adalah Alat Penting

Fungsi adalah alat penting dalam pemrograman. Mereka membantu Kamu mengorganisir, menghemat waktu, dan mengurangi kesalahan. Jadi, selalu ingat untuk menjadikan fungsi sebagai sahabat Kamu dalam petualangan pemrograman.

## Argumen dalam Fungsi

Fungsi juga dapat menerima argumen yang memungkinkan Kamu mengirim data ke dalam fungsi. Misalnya, Kamu ingin membuat fungsi untuk menghitung hasil penambahan dua angka. Berikut contohnya:

```

<?php
function tambah($angka1, $angka2) {
    $hasil = $angka1 + $angka2;
    return $hasil;
}
?>

```

Fungsi `tambah` di atas menerima dua argumen, `\$angka1` dan `\$angka2`, dan mengembalikan hasil penjumlahan keduanya.

## Mengembalikan Nilai

Kamu mungkin perhatikan pernyataan `return` dalam contoh di atas. Ini digunakan untuk mengembalikan nilai dari fungsi. Kamu dapat menangkap nilai yang dikembalikan dari fungsi dan menggunakannya di tempat lain dalam program Kamu.

```

<?php
$hasil_penjumlahan = tambah(5, 3); // Memanggil fungsi dan menyimpan hasilnya.
echo "Hasil penjumlahan: $hasil_penjumlahan"; // Ini akan mencetak "Hasil penjumlahan: 8".
?>

```

## Fungsi Bawaan (Built-in Functions)

Selain membuat fungsi sendiri, PHP juga dilengkapi dengan sejumlah fungsi bawaan yang sangat

berguna. Sebagai contoh, fungsi `strlen()` digunakan untuk menghitung panjang sebuah string, dan fungsi `date()` digunakan untuk menampilkan tanggal dan waktu.

```
~>
<?php
    $kalimat = "Halo, dunia!";
    $panjang_kalimat = strlen($kalimat); // Menghitung panjang string.
    $tanggal_sekarang = date("d/m/Y"); // Mendapatkan tanggal hari ini.
?>
```

Fungsi bawaan ini sangat membantu dalam pemrograman sehari-hari.

## Kesimpulan

Fungsi adalah alat penting dalam pemrograman. Mereka memungkinkan Kamu untuk mengorganisir kode, menghemat waktu, dan mengurangi kesalahan. Kamu dapat membuat fungsi Kamu sendiri, memberikan argumen, dan mengembalikan nilai. Juga, Kamu dapat menggunakan berbagai fungsi bawaan yang sudah disediakan oleh bahasa pemrograman.



Selamat bermain-main dengan fungsi! Mereka adalah dasar yang kuat dalam memahami pemrograman.

## Sub-bab 4.2 Membuat dan Memanggil Fungsi

Halo kembali, teman-teman! Sekarang, setelah kita tahu apa itu fungsi, mari kita pelajari cara membuat dan memanggil fungsi di PHP.

### Membuat Fungsi

Untuk membuat fungsi, Kamu perlu menentukan nama fungsi, argumen (opsional), dan kode yang akan dijalankan ketika fungsi dipanggil. Contoh pembuatan fungsi sederhana:

```
<?php
function sapa() {
    echo "Halo, dunia!";
}
?>
```

Fungsi di atas memiliki nama `sapa` dan ketika dipanggil, ia akan mencetak "Halo, dunia!" di layar.

Kamu juga dapat membuat fungsi dengan argumen:

```
<?php
function sapa($nama) {
    echo "Halo, $nama!";
}
?>
```

## Memanggil Fungsi

Untuk menggunakan fungsi yang sudah Kamu buat, Kamu hanya perlu memanggilnya. Ini dilakukan dengan menyebutkan nama fungsi, dan jika ada argumen, Kamu harus memberikannya.

```
sapa(); // Memanggil fungsi sapa tanpa argumen.
sapa("John"); // Memanggil fungsi sapa dengan argumen "John".
```

Ketika Kamu memanggil fungsi, PHP akan menjalankan kode di dalam fungsi sesuai dengan definisinya.

## Nilai Kembali (Return Value)

Fungsi juga bisa mengembalikan nilai. Contoh:

```
<?php
function tambah($angka1, $angka2) {
    $hasil = $angka1 + $angka2;
    return $hasil;
}
?>
```

Kamu dapat menangkap nilai yang dikembalikan dari fungsi tersebut:

```
$hasil_penjumlahan = tambah(5, 3); // Memanggil fungsi dan menyimpan hasilnya.
```

## Fungsi Bawaan (Built-in Functions)

Selain membuat fungsi Kamu sendiri, PHP sudah menyediakan banyak fungsi bawaan yang sangat berguna. Contohnya, fungsi `strlen()` digunakan untuk menghitung panjang sebuah string.

```
$kalimat = "Halo, dunia!";
$panjang_kalimat = strlen($kalimat); // Menghitung panjang string.
```

## Menggunakan Fungsi Bawaan

Untuk menggunakan fungsi bawaan atau yang telah ada, Kamu hanya perlu memanggilnya seperti fungsi biasa:

```
$hasil = nama_fungsi(argumen); // Contoh pemanggilan fungsi bawaan.
```

## Variabel dalam Fungsi

Variabel dalam fungsi biasanya bersifat lokal. Ini berarti variabel yang Kamu buat dalam fungsi hanya bisa diakses di dalam fungsi tersebut.

Contoh:

```
<?php
function hitungLuas($panjang, $lebar) {
    $luas = $panjang * $lebar;
    return $luas;
}

$hasil = hitungLuas(5, 3);
echo "Luas persegi panjang adalah: $hasil";
?>
```

*Gambar 44 Variabel dalam Fungsi*

Variabel `$luas` adalah variabel lokal yang hanya berlaku di dalam fungsi `hitungLuas`.

## Variabel Global

Variabel yang didefinisikan di luar fungsi dapat diakses di mana saja dalam program Kamu. Ini disebut variabel global.

```
$pesan = "Halo, dunia!"; // Variabel global

function sapa() {
    global $pesan; // Menggunakan variabel global
    echo $pesan;
}

sapa(); // Memanggil fungsi untuk mencetak pesan.
```

## Pentingnya Komentar

Ketika Kamu membuat program yang lebih besar, komentar adalah alat penting untuk menjelaskan apa yang dilakukan oleh fungsi. Ini membantu Kamu dan orang lain yang membaca kode Kamu memahami tujuan fungsi tersebut.

```
function hitung($a, $b) {
    // Fungsi ini digunakan untuk menghitung sesuatu.
    $hasil = $a + $b;
    return $hasil;
}
```

## Contoh Lebih Lanjut

Mari kita lihat contoh lebih kompleks di mana kita menggunakan fungsi untuk menghitung total belanja dan mengaplikasikan diskon:

```
function hitungTotalBelanja($harga, $jumlah) {
    $total = $harga * $jumlah;
    return $total;
}

function berikanDiskon($total, $diskon) {
    $diskon = $total * $diskon;
    $total_diskon = $total - $diskon;
    return $total_diskon;
}

$harga_barang = 100;
$jumlah_barang = 5;
$diskon_persen = 0.1; // 10% diskon

$total_harga = hitungTotalBelanja($harga_barang, $jumlah_barang);
$total_setelah_diskon = berikanDiskon($total_harga, $diskon_persen);

echo "Total belanja: $total_harga";
echo "Total setelah diskon: $total_setelah_diskon";
```

Dengan penggunaan fungsi, kode kita menjadi lebih mudah dimengerti dan dapat diubah dengan mudah.

## Kesimpulan

Membuat dan memanggil fungsi adalah salah satu konsep dasar dalam pemrograman. Fungsi memungkinkan kita untuk mengorganisir kode, menghemat waktu, dan mengurangi kesalahan. Juga,

variabel lokal dan global adalah penting dalam pemahaman cara data disimpan dan diakses dalam program. Terus belajar dan eksplorasi, teman-teman!

### Sub-bab 4.3 Parameter dan Argumen

Hai, teman-teman! Sekarang kita akan membahas lebih dalam tentang parameter dan argumen dalam fungsi. Jadi, apa itu parameter dan argumen? Mari kita bahas.

#### Parameter dalam Fungsi

Parameter adalah variabel yang didefinisikan dalam tkamu kurung saat Kamu membuat fungsi. Mereka bertindak sebagai "input" untuk fungsi tersebut. Parameter digunakan untuk mengirim data ke dalam fungsi sehingga fungsi dapat bekerja dengan data yang diberikan.

Contoh:

```
function sapa($nama) { // $nama adalah parameter
    echo "Halo, $nama!";
}
```

Dalam contoh di atas, ` \$nama ` adalah parameter fungsi `sapa`.

## Argumen saat Memanggil Fungsi

Argumen adalah nilai yang Kamu berikan saat memanggil fungsi. Argumen digunakan untuk mengisi nilai dari parameter yang didefinisikan dalam fungsi.

```
sapa("John"); // "John" adalah argumen yang dikirim ke parameter $nama dalam fungsi.
```

## Parameter yang Lebih dari Satu

Kamu dapat memiliki lebih dari satu parameter dalam fungsi. Contoh:

```
function tambah($angka1, $angka2) {  
    $hasil = $angka1 + $angka2;  
    return $hasil;  
}
```

*Gambar 45 Parameter lebih dari Satu*

Di sini, kita memiliki dua parameter, ` \$angka1 ` dan ` \$angka2 ` . Saat memanggil fungsi, kita harus memberikan dua argumen sesuai dengan jumlah parameter yang diharapkan.



```
$hasil = tambah(5, 3); // Memanggil fungsi dengan dua argumen.
```

## Argumen dan Nilai Default

Dalam PHP, Kamu dapat memberikan nilai default untuk parameter. Ini berarti jika argumen tidak diberikan saat memanggil fungsi, nilai default akan digunakan.

Contoh:

```
function salam($nama = "Teman") {  
    echo "Halo, $nama!";  
}
```

Dalam fungsi di atas, jika kita memanggilmnya tanpa argumen, seperti `salam()`, akan mencetak "Halo, Teman!" karena "Teman" adalah nilai default.

## Contoh Lain

Mari lihat contoh di mana kita menghitung volume balok dengan fungsi dan parameter:

```
function hitungVolume($panjang, $lebar, $tinggi) {  
    $volume = $panjang * $lebar * $tinggi;  
    return $volume;  
}
```

Kemudian kita memanggilnya dengan memberikan argumen:

```
$panjang = 5;  
$lebar = 3;  
$tinggi = 2;  
$hasil = hitungVolume($panjang, $lebar, $tinggi);
```

## Kesimpulan

Sekarang kita tahu bahwa parameter adalah variabel yang digunakan dalam fungsi untuk menerima input, dan argumen adalah nilai yang diberikan saat memanggil fungsi. Dengan pemahaman tentang parameter dan argumen, Kamu dapat membuat fungsi yang lebih dinamis dan berguna. Teruslah belajar, teman!

## **Sub-bab 4.4 Rekursi (Pemanggilan Fungsi Diri Sendiri)**

Hai, teman-teman! Sekarang, mari kita bicarakan tentang sesuatu yang menarik: rekursi. Rekursi adalah ketika sebuah fungsi memanggil dirinya sendiri. Ini seperti dongeng yang menceritakan dirinya sendiri!

### **Apa itu Rekursi?**

Rekursi adalah teknik di mana sebuah fungsi dapat memanggil dirinya sendiri. Ini sering digunakan untuk menyelesaikan masalah yang dapat dibagi menjadi masalah yang lebih kecil dengan cara yang serupa. Pemahaman rekursi dapat memungkinkan Kamu menyelesaikan berbagai masalah yang rumit.

### **Contoh Sederhana: Faktorial**

Mari kita lihat contoh yang sederhana: menghitung faktorial. Faktorial dari suatu angka adalah hasil perkalian dari semua angka dari 1 hingga angka tersebut. Kita dapat menggunakan rekursi untuk menghitung faktorial.

```
function hitungFaktorial($n) {  
    if ($n <= 1) {  
        return 1; // Basis: faktorial dari 0 dan 1 adalah 1  
    } else {  
        return $n * hitungFaktorial($n - 1); // Rekursi  
    }  
}  
  
$hasil = hitungFaktorial(5); // Hasilnya adalah 5! = 120
```

*Gambar 46 Rekursi untuk Menghitung Faktorial*

Jadi, bagaimana ini bekerja? Ketika kita memanggil `hitungFaktorial(5)`, fungsi tersebut memanggil dirinya sendiri dengan argumen `$n - 1` (yaitu, `hitungFaktorial(4)`), dan proses ini berlanjut hingga mencapai basis (ketika `$n` adalah 1 atau kurang) di mana hasilnya adalah 1.

### **Catatan Penting**

1. Pastikan selalu ada kondisi basis yang akan menghentikan rekursi. Tanpa kondisi ini, rekursi akan berlanjut selamanya!
2. Gunakan rekursi dengan bijak. Terdapat masalah yang lebih baik dipecahkan dengan pendekatan non-rekursif.

## Contoh Lain: Penghitung Mundur

Mari kita lihat contoh lain. Kita ingin membuat penghitung mundur sederhana dengan rekursi.

```
function hitungMundur($n) {
    if ($n <= 0) {
        echo "Blastoff!";
    } else {
        echo $n . ", ";
        hitungMundur($n - 1);
    }
}

hitungMundur(5); // Hasilnya adalah "5, 4, 3, 2, 1, Blastoff!"
```

*Gambar 47 Rekursi untuk Penghitung Mundur*

Dalam contoh ini, fungsi `hitungMundur` mencetak nomor secara mundur dari `\$n` ke 1, kemudian mencetak "Blastoff!".

## Kesimpulan

Rekursi adalah konsep yang kuat dalam pemrograman. Ini dapat digunakan untuk memecahkan masalah yang kompleks dengan cara yang elegan dan seringkali lebih mudah dimengerti. Ingat selalu kondisi basis dan jangan biarkan rekursi berjalan tanpa henti! Semangat belajar, teman-teman!

# **BAB 5**

## **OBJECT-ORIENTED PROGRAMMING**

### **(PEMROGRAMAN BERORIENTASI OBJEK)**

#### **Sub-bab 5.1 Konsep Dasar OOP**

Halo, teman-teman! Kita telah menjalani perjalanan pemrograman yang menarik, dan sekarang kita akan memasuki dunia OOP atau Pemrograman Berorientasi Objek. Ini adalah konsep dasar yang sangat penting, dan kita akan menjelaskannya dengan bahasa yang mudah dipahami.

#### **Apa Itu Pemrograman Berorientasi Objek (OOP)?**

OOP adalah paradigma pemrograman di mana program dibangun menggunakan objek. Nah, objek apa itu? Objek adalah entitas yang memiliki atribut (data) dan metode (fungsi). Ini seperti membuat model dunia nyata dalam kode.

#### **Contoh Sederhana: Class dan Objek**

Mari kita lihat contoh sederhana: pemrograman berorientasi objek dalam PHP. Dalam OOP, kita bekerja dengan kelas dan objek.

**Class** adalah cetak biru (blueprint) yang mendefinisikan atribut dan metode yang akan dimiliki oleh objek. Ini adalah ide atau konsep dari objek itu sendiri. Misalnya, kita bisa memiliki class "Mobil" yang mendefinisikan atribut seperti warna, merek, dan metode seperti "Maju"

```
class Mobil {  
    public $warna;  
    public $merek;  
  
    public function maju() {  
        echo "Mobil berwarna " . $this->warna . " bergerak maju.";  
    }  
}
```

**Objek** adalah instance (salinan konkret) dari kelas. Ini adalah representasi nyata dari konsep yang didefinisikan oleh kelas.

```
$mobil1 = new Mobil();  
$mobil1->warna = "Merah";  
$mobil1->merek = "Toyota";
```

Dalam contoh di atas, kita telah membuat objek bernama `$mobil1` yang merupakan instance dari kelas "Mobil" dan mengatur atributnya.

**Metode** adalah fungsi yang terkait dengan objek. Mereka memungkinkan objek untuk melakukan tindakan atau operasi tertentu.

```
$mobil1->maju(); // Output: Mobil berwarna Merah bergerak maju.
```

## Keuntungan OOP

- **Pemisahan Kode**, OOP memungkinkan kita untuk memisahkan bagian-bagian kode yang berbeda ke dalam objek, sehingga lebih mudah dimengerti dan dikelola.
- **Reusabilitas**, Kode yang ditulis dalam OOP cenderung lebih dapat digunakan kembali, sehingga kita tidak perlu menulis ulang kode yang serupa.
- **Abstraksi**, OOP memungkinkan kita untuk menggambarkan konsep dunia nyata dalam kode, yang dapat membuat pemahaman program menjadi lebih mudah.



## Kesimpulan

Pemrograman Berorientasi Objek adalah konsep penting dalam dunia pemrograman. Ini memungkinkan kita untuk memodelkan dunia nyata dalam kode, yang membuat pembangunan dan pemeliharaan program menjadi lebih efisien. Tetap semangat belajar, teman-teman!

## Sub-bab 5.2 Class dan Objek

Halo lagi, teman-teman! Sekarang kita akan lebih mendalam tentang dua konsep penting dalam pemrograman berorientasi objek: **class** (kelas) dan **object** (objek). Ini seperti membuat instruksi dan menciptakan benda nyata dari instruksi tersebut!

### Apa Itu Class dan Objek?

- **Class** adalah seperti resep atau blueprint yang digunakan untuk membuat objek. Kelas mendefinisikan atribut (variabel) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat berdasarkan kelas tersebut. Ini seperti membuat pola yang akan diikuti saat menciptakan objek.

- **Objek** adalah instansi konkret yang dibuat dari suatu kelas. Objek memiliki atribut dan metode yang sesuai dengan apa yang telah didefinisikan dalam kelas. Ini seperti menciptakan barang nyata dari blueprint yang telah dibuat.

## **Kesimpulan**

Class dan objek adalah dasar dari pemrograman berorientasi objek. Mereka memungkinkan kita untuk menciptakan blueprint dan objek nyata berdasarkan blueprint tersebut. Teruslah belajar, teman-teman!

## **Sub-bab 5.3 Enkapsulasi dan Abstraksi**

Hai teman-teman! Sekarang kita akan membahas dua konsep penting dalam pemrograman berorientasi objek: **enkapsulasi** dan **abstraksi**. Ini seperti mengemas dan menyederhanakan informasi agar lebih mudah dimengerti.

### **1. Enkapsulasi: Menyimpan Rapi**

- **Apa itu Enkapsulasi?** Enkapsulasi adalah konsep di mana atribut (variabel) dan metode (fungsi) yang berhubungan dalam suatu kelas dikemas atau "dikemas

rapi" bersama. Ini seperti meletakkan semua benda terkait dalam sebuah kotak agar lebih mudah diatur dan dilindungi.

- **Mengapa Enkapsulasi Penting?** Ini membantu dalam menjaga kode yang bersih dan terorganisir. Objek lain tidak bisa secara langsung mengakses atau mengubah atribut yang "dikapsulasi." Ini menyediakan lapisan perlindungan dan meminimalkan kemungkinan terjadinya kesalahan.

Contoh dalam PHP:

```
class Mobil {
    private $kecepatan;

    public function setKecepatan($nilai) {
        // Kita bisa menambahkan validasi di sini
        if ($nilai > 0) {
            $this->kecepatan = $nilai;
        }
    }

    public function getKecepatan() {
        return $this->kecepatan;
    }
}
```

*Gambar 48 Enkapsulasi*

Dalam contoh di atas, kita menggunakan kata kunci ``private`` untuk mengenkapsulasi atribut ``$kecepatan``. Kemudian, kita menyediakan metode ``setKecepatan`` dan ``getKecepatan`` untuk mengatur dan mendapatkan nilai atribut tersebut.

## 2. Abstraksi: Menyederhanakan

- **Apa itu Abstraksi?** Abstraksi adalah proses menyederhanakan kompleksitas dengan cara menyembunyikan rincian yang tidak penting dan menyorot hal-hal penting. Ini seperti membaca ringkasan buku daripada seluruh teks.

- **Mengapa Abstraksi Penting?** Abstraksi membuat kode lebih mudah dimengerti. Dalam pemrograman, kita tidak perlu tahu seluruh detail suatu objek atau metode, tetapi hanya informasi yang relevan.

Contoh dalam PHP:

```

class Mobil {
    private $warna;
    private $merek;

    public function informasiSingkat() {
        return "Ini adalah mobil " . $this->warna . " merek " . $this->merek
        . " ";
    }
}

```

*Gambar 49 Abstraksi*

Dalam contoh di atas, kita membuat metode `informasiSingkat` yang memberikan ringkasan informasi tentang mobil. Ini lebih mudah dibaca daripada mencantumkan setiap atribut secara terpisah.

## **Kesimpulan**

Enkapsulasi dan abstraksi adalah dua konsep penting dalam pemrograman berorientasi objek. Mereka membantu dalam menjaga kode yang bersih, mudah dimengerti, dan terorganisir dengan baik. Teruslah belajar, teman-teman!

## **Sub-bab 5.4 Pewarisan dan Polimorfisme**

Hai teman-teman! Sekarang kita akan mempelajari dua konsep keren dalam pemrograman berorientasi objek: **pewarisan** dan **polimorfisme**. Ini

seperti mewarisi bakat dari orang tua dan bisa berubah menjadi apa pun yang kita inginkan!

## **1. Pewarisan: Menurunkan Sifat**

- **Apa itu Pewarisan?** Pewarisan adalah konsep di mana kita bisa membuat kelas baru yang "mewarisi" atribut dan metode dari kelas yang sudah ada. Ini seperti anak yang mewarisi sifat-sifat dari orang tuanya.

- **Mengapa Pewarisan Penting?** Ini membantu kita dalam membuat hierarki kelas yang lebih terstruktur. Kita bisa memiliki kelas dasar yang memiliki atribut dan metode umum, dan kemudian membuat kelas-kelas turunan yang menambahkan fitur tambahan.

Contoh dalam PHP:

```

class Hewan {
    public $nama;

    public function bersuara() {
        return "Bunyi umum hewan.";
    }
}

class Kucing extends Hewan {
    public function bersuara() {
        return "Meow!";
    }
}

$kucing = new Kucing();

```

*Gambar 50 Polimorfisme*

Dalam contoh di atas, kita memiliki kelas "Hewan" yang memiliki atribut "nama" dan metode "bersuara." Kemudian, kita membuat kelas "Kucing" yang mewarisi dari kelas "Hewan" dan mengganti metode "bersuara."

## **2. Polimorfisme: Berubah Menjadi Apa yang Dibutuhkan**

- **Apa itu Polimorfisme?** Polimorfisme adalah konsep di mana objek dari kelas yang berbeda dapat merespons metode dengan cara yang berbeda. Ini seperti mengganti pakaian sesuai dengan cuaca.

- **Mengapa Polimorfisme Penting?** Polimorfisme memungkinkan kita untuk menulis kode yang lebih fleksibel. Kita dapat menggunakan metode yang sama dengan objek berbeda dan mendapatkan hasil yang sesuai dengan jenis objek tersebut.

Contoh dalam PHP:

```
function bersuara($hewan) {  
    return $hewan->bersuara();  
}  
  
$kucing = new Kucing();  
$burung = new Burung();  
  
$hasil1 = bersuara($kucing); // Hasilnya adalah "Meow!"  
$hasil2 = bersuara($burung); // Hasilnya adalah "Tweet!"  
?>
```

Dalam contoh di atas, kita memiliki fungsi `bersuara` yang menerima objek hewan sebagai argumen. Meskipun kita memanggilnya dengan objek berbeda (kucing dan burung), hasilnya disesuaikan dengan jenis objek tersebut.

## Kesimpulan



Pewarisan dan polimorfisme adalah konsep yang memungkinkan kita untuk membuat kode yang lebih terstruktur, fleksibel, dan mudah dimengerti. Mereka memberikan "kekuatan ekstra" dalam dunia pemrograman berorientasi objek. Teruslah belajar, teman-teman!

## **BAB 6**

### **PENANGANAN ERROR DAN DEBUGGING**

#### **Sub-bab 6.1 Error dan Exception**

Hai teman-teman! Kita semua tahu bahwa dalam pemrograman, terkadang hal tidak berjalan sesuai rencana. Saat itulah kita berbicara tentang **error** dan **exception**. Tapi jangan khawatir, kita akan menjelaskan semuanya dengan cara yang mudah dimengerti.

##### **1. Error: Kesalahan dalam Kode**

- **Apa itu Error?** Error adalah kesalahan dalam kode yang membuat program tidak berjalan dengan benar. Ini bisa terjadi saat kita salah mengetik, menggunakan sintaks yang salah, atau melakukan hal-hal aneh dalam kode kita.

- **Mengapa Error Penting?** Error membantu kita dalam menemukan masalah dalam kode kita. Mereka adalah petunjuk bahwa ada sesuatu yang perlu diperbaiki.

Contoh dalam PHP:

```
$angka = 10 / 0; // Ini akan menghasilkan error "Division by zero."
```

Dalam contoh di atas, kita mencoba membagi angka dengan 0, yang tidak bisa dilakukan, dan ini menghasilkan error.

## 2. Exception: Tanggapan atas Error

- **Apa itu Exception?** Exception adalah respons yang diberikan oleh program saat menemui error. Ini adalah cara program mengatasi kesalahan. Kita bisa membuat "instruksi darurat" tentang apa yang harus dilakukan saat terjadi error.

- **Mengapa Exception Penting?** Exception membantu kita mengendalikan kesalahan dengan lebih baik. Mereka memungkinkan program untuk menjalankan kode

cadangan atau memberikan pesan kesalahan yang lebih informatif.

Contoh dalam PHP:

```
try {  
    $angka = 10 / 0; // Ini akan menghasilkan exception  
} catch (Exception $e) {  
    echo "Terjadi kesalahan: " . $e->getMessage();  
}
```

*Gambar 51 Exception*

Dalam contoh di atas, kita mencoba membagi angka dengan 0, yang menghasilkan exception. Kemudian, kita menangkap exception tersebut dan mencetak pesan kesalahan.

### **3. Mengatasi Error dan Exception**

- **Bagaimana Cara Mengatasi Error?** Mengatasi error dapat melibatkan melihat pesan kesalahan dan menemukan kesalahan dalam kode kita. Itu seperti bermain teka-teki! Kita perlu melacak di mana masalahnya dan kemudian memperbaikinya.

- **Bagaimana Cara Mengatasi Exception?** Untuk mengatasi exception, kita bisa menggunakan blok `try-catch``. Ini adalah cara untuk mengeksekusi kode yang mungkin menghasilkan exception, dan jika exception terjadi, kita dapat menangani situasinya.

Contoh dalam PHP:

```
try {  
    $angka = 10 / 0;  
} catch (Exception $e) {  
    echo "Terjadi kesalahan: " . $e->getMessage();  
}
```

Dalam contoh di atas, kita mencoba melakukan pembagian yang menghasilkan exception. Tetapi kita menangani exception ini dengan mencetak pesan kesalahan yang bermanfaat.

#### **4. Mencegah Error dan Exception**

- **Bagaimana Cara Mencegah Error?** Untuk mencegah error, sangat penting untuk menulis kode yang teliti dan memeriksa tipe data dan perhitungan yang kita lakukan.

- **Bagaimana Cara Mencegah Exception?** Untuk mencegah exception, kita bisa menggunakan pengujian dan validasi data sebelum menjalankan operasi yang berpotensi menghasilkan exception. Ini adalah cara untuk memastikan data yang kita olah sudah benar.

Contoh dalam PHP:

```
$angka = 10;
if ($angka != 0) {
    $hasil = 10 / $angka;
} else {
    echo "Tidak dapat membagi dengan 0.";
}
```

Dalam contoh di atas, kita memeriksa apakah nilai ` \$angka ` adalah 0 sebelum melakukan pembagian. Ini mencegah exception terjadi.

## Kesimpulan

Error dan exception adalah bagian dari proses belajar pemrograman. Mereka membantu kita memahami,

menangani, dan mencegah masalah dalam kode kita. Ingatlah, tidak ada yang bisa menjadi pemrogram yang hebat tanpa pernah menghadapi kesalahan. Teruslah belajar dan jangan takut untuk mencoba hal baru!

## **Sub-bab 6.2 Handling Exceptions**

Hai teman-teman! Sekarang, mari kita bicarakan tentang penanganan exception, yang merupakan cara keren untuk mengatasi kesalahan dalam kode Kamu. Exception handling memungkinkan kita untuk mengendalikan situasi ketika sesuatu tidak berjalan dengan baik, sehingga program kita bisa tetap berjalan dengan mulus.

### **1. Menggunakan Blok Try-Catch**

- **Apa itu Blok Try-Catch?** Blok try-catch adalah alat utama dalam menangani exception. Dalam blok ``try``, kita menempatkan kode yang mungkin menghasilkan exception. Blok ``catch`` adalah di mana kita menangani exception tersebut.

- **Mengapa Blok Try-Catch Penting?** Blok try-catch memungkinkan program untuk berjalan tanpa terhenti oleh exception. Sebaliknya, jika exception terjadi, kita bisa memberikan respons yang sesuai, seperti mencetak pesan kesalahan.

Contoh dalam PHP:

```
try {
    $hasil = 10 / 0; // Mungkin akan menghasilkan exception
} catch (Exception $e) {
    echo "Terjadi kesalahan: " . $e->getMessage();
}
```

## 2. Blok Finally

- **Apa itu Blok Finally?** Kadang-kadang, ada tindakan yang harus diambil terlepas dari apakah exception terjadi atau tidak. Ini di mana blok `finally` masuk.

- **Mengapa Blok Finally Penting?** Blok finally memastikan bahwa tindakan tertentu selalu dilakukan, bahkan jika exception terjadi. Misalnya, kita bisa membersihkan sumber daya yang digunakan oleh program.

Contoh dalam PHP:

```
try {
    // Kode yang mungkin menghasilkan exception
} catch (Exception $e) {
    // Penanganan exception
} finally {
    // Tindakan yang selalu dilakukan
}
```

### 3. Membuang (Throwing) Exception

- **Apa itu Membuang Exception?** Terkadang kita ingin menyatakan bahwa terjadi kesalahan di dalam fungsi kita. Ini disebut "membuang" exception.

- **Mengapa Membuang Exception Penting?** Dengan melemparkan exception, kita memberi tahu pemanggil fungsi bahwa ada masalah. Ini memungkinkan pemanggil untuk menangani kesalahan tersebut.

Contoh dalam PHP:

```
function bagi($angka1, $angka2) {
    if ($angka2 == 0) {
        throw new Exception("Tidak dapat membagi dengan 0.");
    }
    return $angka1 / $angka2;
}
```

### 4. Mengatasi Berbagai Jenis Exception



Ketika kita menangani exception, kita harus tahu bahwa ada berbagai jenis exception yang bisa terjadi. Mari kita bahas beberapa jenis umumnya:

- **Exception Biasa**, Ini adalah jenis exception yang umum dan mungkin terjadi dalam situasi sehari-hari. Misalnya, ketika kita mencoba membagi angka dengan 0, sebuah exception akan terjadi. Dalam kasus ini, kita dapat menggunakan blok try-catch untuk menangani exception tersebut.

- **Custom Exception**, Kadang-kadang, kita ingin membuat exception kustom sesuai dengan kebutuhan kita. Misalnya, jika kita menulis program untuk mengelola inventaris, kita bisa membuat exception kustom seperti "Barang tidak ditemukan." Ini memungkinkan kita memberikan pesan yang lebih informatif kepada pengguna.

- **Error Handling**, Kadang-kadang, kita ingin menangani error tertentu, bukan exception. Error biasanya menunjukkan kesalahan dalam kode yang sulit diperbaiki.

Misalnya, jika ada kesalahan sintaksis dalam kode PHP, kita bisa menangani error tersebut.

## **5. Menyusun Penanganan Exception yang Baik**

- **Jangan Gunakan Exception untuk Aliran Kontrol**, Exception seharusnya digunakan untuk menangani kesalahan dan situasi tak terduga. Tidak seharusnya digunakan untuk mengontrol aliran program secara normal. Ini akan membuat kode Kamu sulit dibaca dan dipahami.

- **Gunakan Pesan Kesalahan yang Informatif**, Ketika Kamu menangani exception, pastikan pesan kesalahan yang dihasilkan informatif. Ini membantu dalam pemecahan masalah dan memahami apa yang salah.

- **Tes dan Debug**, Sebelum Kamu menyelesaikan kode Kamu, lakukan pengujian untuk menemukan potensi exception dan pastikan mereka ditangani dengan benar. Gunakan alat debugging jika diperlukan.

## **Kesimpulan**

Penanganan exception adalah keterampilan yang sangat penting dalam pemrograman. Ini memungkinkan kita untuk mengendalikan kesalahan dan membuat program yang lebih kuat dan kamul. Dengan memahami berbagai jenis exception dan menulis penanganan yang baik, Kamu dapat menjadi pemrogram yang lebih baik. Jadi, jangan ragu untuk mencoba, belajar, dan terus meningkatkan keterampilan penanganan exception Kamu!

## **Sub-bab 6.3 Debugging Tools dan Teknik**

Halo teman-teman! Sekarang kita akan membahas alat dan teknik debugging. Debugging adalah cara kita menemukan dan memperbaiki kesalahan dalam kode kita. Ini adalah keterampilan yang sangat penting dalam pemrograman, jadi mari kita mulai!

### **1. Pesan Debugging (echo dan print)**

- **Apa itu Pesan Debugging?** Pesan debugging adalah cara termudah untuk memeriksa nilai variabel atau mengetahui alur eksekusi program. Kamu bisa

menggunakan perintah ``echo`` atau ``print`` dalam PHP untuk mencetak pesan atau nilai ke layar.

- **Mengapa Ini Berguna?** Dengan menambahkan pesan debugging pada kode Kamu, Kamu bisa melihat apa yang terjadi di dalam program Kamu. Misalnya, Kamu bisa mencetak nilai variabel atau pesan "Saya ada di sini" pada titik-titik tertentu dalam kode Kamu.

Contoh dalam PHP:

```
$angka = 42;  
echo "Nilai angka adalah: " . $angka;
```

## 2. Penggunaan ``var_dump`` dan ``print_r``

- **Apa itu ``var_dump`` dan ``print_r``?** Ini adalah fungsi bawaan dalam PHP yang memungkinkan Kamu untuk mencetak struktur data yang lebih kompleks seperti array atau objek. ``var_dump`` memberikan informasi yang lebih rinci, sementara ``print_r`` memberikan tampilan yang lebih mudah dibaca.

- **Mengapa Ini Berguna?** Ketika Kamu bekerja dengan struktur data kompleks, fungsi ini membantu Kamu memahami bagaimana data tersebut tersusun dan memudahkan pemecahan masalah.

Contoh dalam PHP:

```
$data = array("apel", "jeruk", "mangga");  
var_dump($data);
```

### 3. Menggunakan Debugger

- **Apa itu Debugger?** Debugger adalah alat khusus yang digunakan untuk menganalisis kode Kamu selangkah demi selangkah. Kamu bisa menetapkan titik henti (breakpoint) di dalam kode dan memeriksa nilai variabel saat program berhenti di breakpoint.

- **Mengapa Ini Berguna?** Debugger sangat kuat untuk mengatasi masalah kompleks dalam kode. Ini memungkinkan Kamu melihat alur eksekusi program secara mendalam.

### 4. Menggunakan Komentar

- **Apa itu Komentar?** Komentar adalah pesan yang Kamu sisipkan di dalam kode Kamu, yang tidak akan dieksekusi oleh program. Ini adalah cara bagus untuk menjelaskan apa yang dilakukan oleh bagian kode tertentu.

- **Mengapa Ini Berguna?** Komentar membantu Kamu dan orang lain yang melihat kode Kamu untuk memahami niat dan fungsionalitas kode.

Contoh dalam PHP:

```
// Ini adalah komentar. Ini bagian dari kode yang tidak akan dieksekusi.
```

## 5. Menggunakan Log

- **Apa itu Log?** Log adalah catatan yang dibuat oleh program Kamu saat berjalan. Kamu bisa memasukkan pesan log ke dalam kode Kamu untuk memantau apa yang terjadi selama eksekusi.

- **Mengapa Ini Berguna?** Log membantu Kamu melacak apa yang terjadi dalam aplikasi Kamu, terutama saat aplikasi berjalan di lingkungan produksi. Kamu bisa menggunakan alat seperti ``error_log`` dalam PHP untuk mencatat pesan.

Contoh dalam PHP:

```
error_log("Ini adalah pesan log.");
```

## 6. Penggunaan Try-Catch

- **Apa itu Try-Catch?** Try-Catch adalah mekanisme yang memungkinkan Kamu menangani exception (kesalahan) dengan lebih elegan. Kamu mengelilingi kode yang mungkin menyebabkan exception dengan blok `try`. Jika exception terjadi, Kamu bisa menangani (atau "menangkap") exception tersebut dalam blok `catch`.

- **Mengapa Ini Berguna?** Try-Catch membantu mencegah program Kamu berhenti saat exception terjadi. Sebaliknya, Kamu bisa memberikan penanganan yang sesuai, seperti memberikan pesan kesalahan kepada pengguna.

Contoh dalam PHP:

```
try {  
    // Kode yang mungkin menyebabkan exception  
    $hasil = 10 / 0;  
} catch (Exception $e) {  
    // Penanganan exception  
    echo "Terjadi kesalahan: " . $e->getMessage();  
}
```

## 7. Menggunakan Xdebug (Alat Debugging untuk PHP)

- **Apa itu Xdebug?** Xdebug adalah alat debugging yang kuat untuk PHP. Ini memungkinkan Kamu untuk menjalankan kode PHP dalam mode debugging dan memberikan alat visual untuk menganalisis kode Kamu.

- **Mengapa Ini Berguna?** Xdebug menyediakan alat yang kuat untuk analisis kode PHP secara mendalam. Ini adalah pilihan yang baik jika Kamu bekerja pada proyek PHP yang kompleks.

## 8. Mencari Bantuan dari Komunitas dan Forum

- **Apa itu Komunitas dan Forum?** Ada banyak komunitas online dan forum di mana para pemrogram berkumpul. Kamu bisa bergabung dengan komunitas PHP atau forum pemrograman untuk mencari bantuan dari sesama pemrogram.



- **Mengapa Ini Berguna?** Terkadang, masalah yang sulit diatasi dapat dipecahkan oleh seseorang yang telah mengalami masalah yang sama. Komunitas dan forum adalah tempat yang bagus untuk bertanya dan berbagi pengetahuan.

## 9. Membaca Dokumentasi Resmi

- **Apa itu Dokumentasi Resmi?** Setiap bahasa pemrograman memiliki dokumentasi resmi yang merinci cara menggunakan bahasa tersebut. Dokumentasi ini seringkali menyertakan contoh kode dan penjelasan.

- **Mengapa Ini Berguna?** Dokumentasi resmi adalah sumber informasi yang paling akurat dan kamul. Saat Kamu mengalami masalah, periksa dokumentasi resmi untuk menemukan solusi.

## Kesimpulan

Debugging adalah keterampilan penting dalam pemrograman. Dengan berbagai alat dan teknik, Kamu dapat dengan mudah menemukan dan memperbaiki kesalahan dalam kode Kamu. Jangan ragu untuk mencoba berbagai metode debugging dan selalu terbuka untuk

belajar lebih banyak. Semakin Kamu berlatih, semakin baik Kamu akan menjadi dalam menemukan kesalahan dalam kode Kamu!

## **BAB 7**

### **PEMROGRAMAN BERBASIS KASUS**

#### **Sub-bab 7.1 Studi Kasus I Pembuatan Aplikasi Sederhana**

Selamat datang di sub-bab ini, di mana Kamu akan memulai perjalanan untuk membangun aplikasi sederhana. Aplikasi ini akan memungkinkan pengguna untuk menghitung hasil penjumlahan dua angka. Meskipun sederhana, ini akan memberi Kamu dasar-dasar dalam memahami bagaimana membangun aplikasi web yang interaktif.

#### **Langkah 1: Persiapan**

Sebelum memulai, pastikan Kamu memiliki lingkungan pengembangan yang sesuai. Kamu akan memerlukan:

- **Editor Kode.** Kamu bisa menggunakan editor teks sederhana seperti Notepad (Windows) atau Visual Studio Code (gratis dan sangat baik).
- **Web Server,** Kamu bisa menggunakan server lokal seperti XAMPP (Windows, macOS, Linux) atau MAMP (macOS) untuk menjalankan aplikasi web Kamu secara lokal.
- **Browser,** Kamu akan memerlukan browser web untuk melihat aplikasi Kamu.

## **Langkah 2: Membuat Struktur Dasar**

1. Buat direktori baru untuk proyek Kamu, misalnya "kalkulator-sederhana".
2. Di dalam direktori ini, buat file dengan nama "index.html". Ini adalah berkas utama aplikasi web Kamu.
3. Buka "index.html" dengan editor teks Kamu dan mulai menulis kode HTML dasar. Inilah tampilan awal aplikasi Kamu.

```
index.html x +
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Kalkulator Sederhana</title>
5 </head>
6 <body>
7   <h1>Selamat datang di Kalkulator Sederhana</h1>
8 </body>
9 </html>
```

### Langkah 3: Membuat Formulir Input

Sekarang, mari tambahkan formulir input untuk memasukkan dua angka yang ingin dijumlahkan.

```
index.html x +
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Kalkulator Sederhana</title>
5 </head>
6 <!-- ... (kode sebelumnya) ... -->
7
8 <body>
9   <h1>Selamat datang di Kalkulator Sederhana</h1>
10  <form>
11    Angka 1: <input type="text" name="angka1"><br>
12    Angka 2: <input type="text" name="angka2"><br>
13    <input type="submit" value="Hitung">
14  </form>
15 </body>
16 </html>
```

## Langkah 4: Menambahkan Logika PHP

Sekarang, kita akan menambahkan kode PHP untuk mengambil input pengguna, menjumlahkannya, dan menampilkan hasilnya.

```
<!-- ... (kode sebelumnya) ... -->

<body>
  <h1>Selamat datang di Kalkulator Sederhana</h1>
  <form method="post">
    Angka 1: <input type="text" name="angka1"><br>
    Angka 2: <input type="text" name="angka2"><br>
    <input type="submit" value="Hitung">
  </form>

  <?php
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $angka1 = $_POST["angka1"];
    $angka2 = $_POST["angka2"];
    $hasil = $angka1 + $angka2;
    echo "Hasil: " . $hasil;
  }
  ?>
</body>
</html>
```

## Langkah 5: Uji Aplikasi Kamu

Sekarang, Kamu dapat membuka browser Kamu dan mengakses file "index.html" di server lokal Kamu (misalnya, <http://localhost/kalkulator-sederhana/index.html>). Kamu akan melihat formulir

input, dan saat Kamu mengisi angka-angka dan mengklik "Hitung," hasil penjumlahan akan ditampilkan.

Inilah langkah awal Kamu dalam membangun aplikasi web sederhana. Kamu telah memahami cara membuat formulir input, mengambil input pengguna, dan melakukan perhitungan sederhana. Teruslah belajar dan menjelajahi lebih lanjut untuk membangun aplikasi yang lebih kompleks!

## **Sub-bab 7.2 Studi Kasus II Pengembangan Game Sederhana**

Pengembangan game sederhana dengan HTML dan PHP memerlukan kreativitas dalam menggabungkan elemen-elemen web dengan logika permainan. Di bawah ini, kami akan membuat game "Tebak Angka" sederhana menggunakan HTML, PHP, dan sedikit JavaScript.

### **Langkah 1: Persiapan**

Pastikan Kamu memiliki teks editor yang nyaman untuk menulis kode, dan Kamu dapat menjalankan kode PHP di server lokal atau hosting web.

## Langkah 2: Struktur Dasar HTML

Buat file HTML dasar untuk tampilan game "Tebak Angka". Kamu dapat mengatur tampilan seperti tombol "Mulai", area untuk menebak angka, dan kotak pesan. Berikut contoh struktur HTML-nya:

```
index.html × +
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Game Tebak Angka</title>
5   <link rel="stylesheet" type="text/css" href="style.css">
6 </head>
7 <body>
8   <h1>Game Tebak Angka</h1>
9   <p>Masukkan tebakan Kamu antara 1 hingga 100:</p>
10  <input type="text" id="tebakan" onkeyup="cekTebakan()">
11  <button id="btnMulai" onclick="mulaiGame()">Mulai</button>
12  <p id="pesan"></p>
13  <script src="script.js"></script>
14 </body>
15 </html>
```

## Langkah 3: Logika Game dengan PHP dan JavaScript

Buat dua file terpisah: "script.js" untuk logika game menggunakan JavaScript dan "game.php" untuk mengelola data permainan dan hasil tebakan. Di bawah ini adalah contoh kode untuk "script.js":

```
script.js x +
script.js
1 let angkaAcak = Math.floor(Math.random() * 100) + 1;
2 let kesempatan = 10;
3
4 function mulaiGame() {
5     angkaAcak = Math.floor(Math.random() * 100) + 1;
6     kesempatan = 10;
7     document.getElementById("btnMulai").disabled = true;
8     document.getElementById("pesan").innerHTML = "";
9 }
10
11 function cekTebakan() {
12     let tebakkan = parseInt(document.getElementById("tebakan").value);
13
14     if (tebakkan === angkaAcak) {
15         document.getElementById("pesan").innerHTML = "Selamat! Kamu berhasil menebak angka " +
16         angkaAcak + ".";
17         document.getElementById("btnMulai").disabled = false;
18     } else if (tebakkan < angkaAcak) {
19         document.getElementById("pesan").innerHTML = "Tebakan terlalu rendah. Kesempatan: " +
20         kesempatan;
21     } else {
22         document.getElementById("pesan").innerHTML = "Tebakan terlalu tinggi. Kesempatan: " +
23         kesempatan;
24     }
25     kesempatan--;
26     if (kesempatan === 0) {
27         document.getElementById("pesan").innerHTML = "Kamu kehabisan kesempatan. Angka yang benar
28         adalah " + angkaAcak + ".";
29         document.getElementById("btnMulai").disabled = false;
30     }
31 }
```

Dengan mengikuti langkah-langkah ini, Kamu dapat membuat game "Tebak Angka" sederhana yang berfungsi menggunakan HTML, PHP, dan JavaScript. Kamu dapat mengembangkan lebih lanjut dengan menambahkan fitur-fitur seperti poin, tingkat kesulitan, atau tampilan yang lebih menarik sesuai dengan keahlian dan kreativitas Kamu.



Selamat mencoba! Pastikan Kamu menjalankan aplikasi ini di lingkungan server yang mendukung PHP, seperti XAMPP atau web hosting yang kompatibel dengan PHP.

## DAFTAR PUSTAKA

- Andri Nofiar. (2020). Pemrograman Dasar. Politeknik  
Kampar.
- Anita Qoriah, Rina Harimurti, Andi Iwan Nurhidayat,  
Asmunin. (2019). Pemrograman Dasar.
- Agung Setiawan. (2021). Modul Struktur Data.
- Triase. (2020). Diktat Edisi Revisi Struktur Data.  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSTAS ISLAM NEGERI SUMATERA UTARA  
MEDAN
- Rudolf Pecinovsky. (2013). OOP – Learn Object Oriented  
Thinking and Programming.
- Joel C. Adams. (2006). OOP and the Janus Principle.
- Muhammad Surya Iksanudin. (2018). Pemrograman  
Berbasis Objek Modern Dengan PHP.
- Jafar Shadiq, Dony Oscar. (2017). Pembelajaran  
Pemrograman Berorientasi Objek (Object  
Oriented Programming) Berbasis Project Based  
Learning.

Muhammad Randyka Rojat. (2022). Pembelajaran Pemrograman Berorientasi Objek (Object Oriented Programming) Berbasis Project Based Learning.

Shapiro, Ehud Yehuda. (1982). ALGORITHMIC PROGRAM DEBUGGING.

Andreas Zeller. (2009). Why Programs Fail: A Guide to Systematic Debugging.

Irvin R. Katz & John R. Anderson. (2009). Debugging: An Analysis of Bug-Location Strategies.

**Penerbit :**

Universitas Islam Kalimantan  
Muhammad Arsyad Al -Banjary

**Alamat :**

Gedung A UPT Publikasi dan Pengelolaan Jurnal  
Universitas Islam Kalimantan  
Muhammad Arsyad Al-Banjary

Jl. Adhyaksa No. 2 Kayutangi  
Banjarmasin, Kalimantan Selatan  
Telepon : 0511 - 3304352  
FAX : 0511

ISBN 978-623-8189-13-7

