BAHASA PEMROGRAMAN DASAR

MENGGUNAKAN PHYTON, C++, JAVA DAN PHP

Supriadi Panggabean, S.Kom., M.Pd., M.Kom.

Supriadi Panggabean, S.Kom., M.Pd., M.Kom.

BAHASA PEMROGRAMAN DASAR MENGGUNAKAN PHYTON, C++, JAVA DAN PHP

Universitas Darunnajah Press

UNDANG-UNDANG REPUBLIK INDONESIA
NOMOR 28 TAHUN 2014 TENTANG HAK CIPTA

LINGKUP HAK CIPTA
Pasal 1

1. Hak Cipta adalah hak eksklusif pencipta yang timbul secara otomatis berdasarkan prinsip deklaratif setelah suatu ciptaan diwujudkan dalam bentuk nyata tanpa mengurangi pembatasan sesuai dengan ketentuan peraturan perundang-undangan.

KETENTUAN PIDANA
Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).
3. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (l) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/ atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah).
4. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah).

BAHASA PEMROGRAMAN DASAR MENGGUNAKAN PHYTON, C++, JAVA DAN PHP

All Right Reserved
Hak Cipta Dilindungi Undang-undang
Hak Penerbitan pada Universitas Darunnajah Press

ISBN: 978-634-04-7828-0

Penulis:
Supriadi Panggabean, S.Kom., M.Pd., M.Kom.

Editor:
Falah Ibrahim Wahid, M.Comm

Lay out
Falah Ibrahim Wahid, M.Comm

Desain sampul:
Supriadi Panggabean, S.Kom., M.Pd., M.Kom.

Cetakan I, 2025

Hak Cipta 2025 pada Penulis
Hak penerbitan pada Universitas Darunnajah Press. Bagi mereka yang ingin memperbanyak sebagian isi buku ini dalam bentuk atau cara apapun harus mendapatkan izin tertulis dari penulis dan penerbit Universitas Darunnajah Press.

Penerbit
Universitas Darunnajah Press
No. Anggota IKAPI : 633/Anggota Luar Biasa/DKI/2024
Jln. Ciledug Raya No. 01. Ulujami Raya, Pesanggrahan
Jakarta Selatan, Provinsi DKI Jakarta
Website: www.press.darunnajah.ac.id
Instagram: @udn_press

KATA PENGANTAR

Buku ini disusun dengan tujuan memberikan panduan praktis dan pemahaman yang mendalam mengenai dasar-dasar pemrograman. Untuk menilai kualitas dan relevansi isi buku ini, penulis telah meminta pendapat dari beberapa ahli di bidang teknologi informasi.

Menurut **Dr. Andi Prasetyo**, seorang pakar teknologi informasi, "Buku ini berhasil menyajikan konsep pemrograman dengan cara yang sistematis dan mudah dipahami. Contoh-contoh praktis yang disediakan memudahkan pembaca untuk mengaplikasikan teori dalam kehidupan nyata, sehingga buku ini sangat layak dijadikan referensi belajar, baik untuk pemula maupun yang sudah berpengalaman."

Selain itu, **Prof. Rina Hartati**, seorang akademisi di bidang Ilmu Komputer, menyatakan bahwa "Buku ini tidak hanya menyampaikan teori, tetapi juga menekankan penerapan konsep melalui studi kasus dan contoh program. Pendekatan seperti ini sangat efektif untuk memperkuat pemahaman pembaca dan meningkatkan keterampilan pemrograman secara nyata."

Pendapat dari para ahli tersebut menjadi motivasi bagi penulis dan pembaca untuk terus mengembangkan pemahaman dan kemampuan di bidang pemrograman. Semoga buku ini dapat memberikan manfaat yang luas dan menjadi referensi yang berguna bagi siapa saja yang ingin mempelajari dunia pemrograman.

PRAKATA

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya buku "Bahasa Pemrograman Dasar Menggunakan Python, C++, Java, dan PHP" ini dapat diselesaikan dengan baik. Buku ini disusun sebagai panduan komprehensif bagi mahasiswa, pelajar, maupun pembelajar mandiri yang ingin memahami konsep dasar pemrograman melalui empat bahasa populer yang banyak digunakan dalam dunia akademik maupun industri.

Perkembangan teknologi informasi yang begitu pesat menuntut kemampuan pemrograman yang tidak hanya bersifat teoretis, tetapi juga aplikatif. Oleh karena itu, buku ini dirancang untuk memberikan pemahaman mulai dari konsep paling dasar—seperti variabel, tipe data, percabangan, perulangan, dan fungsi—hingga penerapan modularisasi dan pembuatan program sederhana. Penyajian materi menggunakan Python, C++, Java, dan PHP bertujuan untuk memberikan wawasan luas mengenai gaya penulisan, struktur, dan paradigma dari berbagai bahasa pemrograman sehingga pembaca dapat memilih dan menguasai bahasa yang sesuai dengan kebutuhan atau minatnya.

Penulis menyadari bahwa penyusunan buku ini tidak terlepas dari dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada rekan-rekan dosen, mahasiswa, serta semua pihak yang telah memberikan masukan, motivasi, dan dorongan sehingga buku ini dapat terselesaikan. Penulis berharap buku ini dapat menjadi bahan ajar yang bermanfaat, baik dalam kegiatan perkuliahan, pelatihan, maupun pembelajaran mandiri.

Akhir kata, penulis sangat terbuka terhadap kritik dan saran demi perbaikan buku ini pada edisi berikutnya. Semoga buku ini dapat memberikan manfaat dan kontribusi dalam meningkatkan kemampuan serta pemahaman pembaca di bidang pemrograman komputer.

Penulis,
Supriadi Panggabean, S.Kom., M.Pd., M.Kom.

DAFTAR ISI

KATA PENGANTAR iv
PRAKATA v
DAFTAR ISI vi
DAFTAR TABEL xi
DAFTAR GAMBAR xii
BAB 1 PENGENALAN PEMROGRAMAN 13
A. Program 13
B. Programmer 14
C. Bahasa Pemrograman 15
D. Pemrograman 17
E. Alasan Belajar Pemrograman 18
F. Peran Pemrograman Dalam Kehidupan 19
BAB 2 DASAR LOGIKA PEMROGRAMAN 22
A. Pendahuluan 22
B. Algoritma 22
C. Pseudocode 25
D. Flowchart 27
E. Logika IF sederhana 28
F. Instalasi Phyton 31
G. Instalasi C++ 34
H. Instalasi Java 37
I. Instalasi PHP 41
J. Struktur Dasar Kode Program 46
K. Aturan Dasar Kode Program 57
BAB 3 VARIABEL DAN TIPE DATA 72
A. Pendahuluan 72
B. Konsep Variabel 72
C. Konstanta (Nilai Tetap) 76

D. Tipe Data 78
E. Operasi String 161
F. Struktur Data Sederhana: List dan Tuple 162
G. Representasi Bilangan 164
BAB 4 OPERATOR DALAM PEMROGRAMAN 166
A. Pendahuluan 166
B. Operator Aritmatika 166
C. Operator Perbandingan 168
D. Operator Logika 170
E. Operator Boolean 172
F. Operator String 174
G. Operator Increment & Decrement 175
H. Operator Bitwise 177
I. Operator Assignment (Penugasan) 179
J. Operator Identitas dan Keanggotaan 180
K. Operator Gabungan Assignment 187
BAB 5 INPUT DAN OUTPUT (I/O) 190
A. Pendahuluan 190
B. Jenis-jenis Input 191
C. Jenis-Jenis Output 193
D. Sintaks Input dan Output 194
E. Format Output 196
F. Validasi Input 199
G. Kesalahan Umum dalam Input–Output 203
H. Contoh Kasus: Program Luas Persegi Panjang 205
BAB 6 PERCABANGAN 207
A. Pendahuluan 207
B. IF Tunggal 207
C. IF-ELSE 209
D. IF-ELIF-ELSE (atau IF-ELSE IF-ELSE) 211
E. SWITCH / CASE 213
F. Studi Kasus 215

BAB 7 PERULANGAN 220
A. Pendahuluan (Jika Dibutuhkan) 220
B. While Loop 220
C. For Loop 222
D. Do-While Loop 223
E. Infinite Loop 225
F. Kendali Looping: Break, Continue, Pass 226
G. Studi Kasus: Pencarian Data dalam List/Array 228
BAB 8 ARRAY / KOLEKSI DATA 233
A. Pendahuluan (Jika Dibutuhkan) 233
B. Konsep Array 233
C. Akses Elemen Array 234
D. Iterasi Array 236
E. Studi Kasus: Daftar Nilai Mahasiswa 238
BAB 9 FUNGSI DAN PROSEDUR 240
A. Pendahuluan 240
B. Konsep Modularisasi 240
C. Fungsi dengan Parameter 241
D. Argument Fungsi 243
E. Scope Variabel (Lokal vs Global) 246
F. Pernyataan Return 248
G. Prosedur (Fungsi Tanpa Return) 249
H. Docstring / Dokumentasi Fungsi 251
I. Studi Kasus: Fungsi Kalkulator Sederhana 252
BAB 10 PEMROGRAMAN BERORIENTASI OBJEK (OOP) 255
A. Pendahuluan 255
B. Konsep Class dan Object 255
C. Atribut & Method 266
D. Encapsulation (Enkapsulasi) 272
E. Inheritance (Pewarisan) dalam OOP 279
F. Polymorphism (Polimorfisme) 283
G. Abstraction (Abstraksi) 287

H. Studi Kasus OOP: Sistem Manajemen Mahasiswa 291
BAB 11 ERROR DAN DEBUGGING 300
A. Pendahuluan 300
B. Jenis-Jenis Error dalam Pemrograman 300
C. Sumber Umum Penyebab Error 302
D. Teknik-Teknik Debugging 303
E. Pentingnya Error dan Debugging dalam OOP 304
F. Proses Debugging yang Sistematis 305
G. Teknik Debugging Sederhana 306
H. Tools Debugging pada Berbagai Bahasa 306
I. Debugging pada Program Berorientasi Objek 307
J. Logging dan Monitoring 308
K. Exception Handling & Custom Exception 308
L. Testing sebagai Pencegah Error 310
M. Debugging pada Multithreading atau Asynchronous 311
N. Dokumentasi Error 311
O. Best Practices Mencegah Error 312
BAB 12 FILE HANDLING (PENGELOLAAN FILE) 313
A. Pengantar File Handling 313
B. Jenis-Jenis File 316
C. Operasi Dasar dalam File Handling 317
D. Membaca File 321
E. Menulis dan Memperbarui File 323
F. File Teks vs File Biner 325
G. Manajemen File dan Direktori 326
H. Exception Handling pada File 329
I. File Handling Berbasis OOP 332
J. Library / Modul File Handling 336
K. File Buffering dan Stream 338
L. Penanganan File di Sistem Berbeda 341
M. Studi Kasus Implementasi 346
N. Best Practices dalam File Handling 349

BAB 13 MODULARISASI & LIBRARY 352
A. Pengantar Modularisasi 352
B. Konsep Modul dan Fungsi 352
C. Ilustrasi Pemanggilan Fungsi Antar Modul 359
D. Library (Pustaka) dalam Pemrograman 362
E. Penggunaan Library Standar 363
F. Penggunaan Library Eksternal 364
G. Modularisasi Proyek Besar 365
BAB 14 STUDI KASUS MINI PROJECT 367
A. Mini Project 1 — Aplikasi Kalkulator 367
B. Mini Project 2 — Program Kasir Sederhana 370
C. Mini Project 3 — Sistem Manajemen Data Mahasiswa 373
DAFTAR PUSTAKA 377
GLOSARIUM 381
BIOGRAFI PENULIS 392

DAFTAR TABEL

Tabel 1. 1 Contoh Penggunaaan Bahasa Pemrograman 16
Tabel 3. 1 Aturan Penamaan Variabel 73
Tabel 3. 2 Deklarasi Variabel pada Berbagai Bahasa 73
Tabel 3. 3 Definisi Konstanta dalam Python, C++, Java, dan PHP 76
Tabel 3. 4 Contoh Tipe Data pada 4 Bahasa Pemrograman 79
Tabel 3. 5 Tipe Data Signed pada C++ 98
Tabel 3. 6 Tipe Data Unsigned pada C++ 98
Tabel 3. 7 Perbedaan Float dan Double pada C++ 102
Tabel 3. 8 Besar Memori Dan Rentang Nilai Tipe pada Java 126
Tabel 3. 9 Perbedaan Float dan Double pada Java 130
Tabel 4. 1 Daftar Operator Aritmatika 167
Tabel 4. 2 Daftar Operator Aritmatika 169
Tabel 4. 3 Daftar Operator Logika 171
Tabel 4. 4 Daftar Operator Aritmatika 177
Tabel 4. 5 Daftar Operator Aritmatika 188
Tabel 5. 1 Cara Menampilkan Output dengan Format 197
Tabel 10. 1 Perbedaan Class dam Object 261
Tabel 10. 2 Access Modifier Public, Protected and Private 273
Tabel 10. 3 Konsep Access Modifier 274
Tabel 10. 4 Manfaat Inheritance 279
Tabel 10. 5 Jenis-jenis Inheritance 282
Tabel 10. 6 Manfaat Inheritance 283
Tabel 10. 7 Manfaat Abstraction 287
Tabel 10. 8 Hubungan Abstraction dan Encapsulation 289
Tabel 10. 9 Konsep OOP yang Tercover dalam Studi Kasus 299
Tabel 12. 1 Mode Pembukaan File dalam Empat Bahasa 319

DAFTAR GAMBAR

Gambar 1. 1 Peran Pemrograman dan Kehidupan 20
Gambar 2. 1 Contoh Flowchart 28
Gambar 10. 1 Diagram Encapsulation 278
Gambar 10. 2 Class Diagram 292
Gambar 12. 1 Diagram umum operasi file 318
Gambar 12. 2 Diagram alur pembukaan file 320
Gambar 12. 3 Diagram alur membaca file 321
Gambar 12. 4 Diagram alur membaca file 323

BAB 1
PENGENALAN PEMROGRAMAN

A. Program

Program adalah kumpulan instruksi atau perintah yang ditulis dalam bahasa pemrograman dan dirancang untuk dijalankan oleh komputer (Andaria, 2025). Instruksi-instruksi ini mengarahkan komputer untuk melakukan tugas-tugas tertentu sesuai dengan tujuan yang diinginkan oleh pembuat program. Bahasa pemrograman yang digunakan bisa bermacam-macam, seperti Python, Java, atau C++, tergantung pada kebutuhan dan platform yang digunakan. Instruksi dalam program harus ditulis secara logis, runtut, dan sesuai dengan aturan sintaksis dari bahasa yang dipilih agar dapat dipahami dan dijalankan oleh komputer tanpa kesalahan.

Program merupakan hasil akhir dari proses yang disebut pemrograman. Dalam dunia nyata, program digunakan untuk berbagai macam tujuan, mulai dari aplikasi yang sangat sederhana seperti kalkulator, hingga sistem yang kompleks seperti software pengelola keuangan perusahaan atau sistem kecerdasan buatan (Anggraini et al., 2025). Program bisa berjalan di berbagai perangkat, mulai dari komputer desktop, laptop, smartphone, hingga perangkat IoT (Internet of Things). Fungsi program adalah untuk membantu pengguna menyelesaikan permasalahan atau mengotomatisasi pekerjaan tertentu, sehingga menjadi lebih cepat, efisien, dan akurat (Panggabean et al., 2023).

Contoh dari program yang sering digunakan dalam kehidupan sehari-hari meliputi Microsoft Word untuk mengetik dokumen, WhatsApp untuk berkomunikasi, dan browser seperti Google Chrome untuk mengakses internet. Di balik setiap program tersebut, terdapat ribuan hingga jutaan defis kode yang bekerja secara terstruktur. Tanpa program, perangkat keras seperti komputer atau ponsel hanya akan menjadi benda

mati yang tidak berguna, karena tidak memiliki instruksi untuk dijalankan. Dengan kata lain, program adalah "jiwa" dari perangkat digital, yang membuatnya mampu berpikir dan bertindak sesuai dengan keinginan pengguna.

B. Programmer

Programmer adalah individu yang memiliki kemampuan untuk merancang, membuat, dan mengembangkan program komputer yang dapat menjalankan fungsi tertentu sesuai kebutuhan pengguna (Monica & Candradewini, 2025). Dalam proses pengembangan perangkat lunak, seorang programmer bertanggung jawab menulis kode menggunakan bahasa pemrograman, menyusun alur logika, serta memastikan setiap bagian kode dapat berjalan secara efisien dan bebas dari kesalahan (Kurniawan & Publisher, 2025). Tugas tersebut tidak hanya berfokus pada penulisan kode semata, tetapi juga mencakup proses pengujian (testing) untuk memastikan program bekerja sesuai spesifikasi, serta melakukan perbaikan atau optimalisasi apabila ditemukan bug atau kekurangan dalam program. Pemahaman yang baik terhadap struktur data, algoritma, dan konsep komputasi menjadi syarat penting agar seorang programmer mampu menghasilkan perangkat lunak yang berkualitas.

Selain kemampuan teknis, pekerjaan seorang programmer juga menuntut keterampilan non-teknis yang tidak kalah penting, khususnya ketika bekerja dalam sebuah tim pengembang. Programmer harus mampu berkolaborasi dengan desainer, analis sistem, manajer proyek, hingga klien untuk memastikan bahwa perangkat lunak yang dibuat benar-benar sesuai dengan kebutuhan pengguna akhir. Kemampuan komunikasi, pemahaman kebutuhan bisnis, serta manajemen waktu menjadi aspek pendukung yang sangat menentukan keberhasilan suatu proyek. Dengan demikian, profesi programmer tidak hanya berpusat pada penulisan kode, tetapi juga pada kemampuan menyelesaikan masalah secara sistematis, bekerja sama dalam tim, dan beradaptasi terhadap perkembangan teknologi yang terus berubah.

C. Bahasa Pemrograman

Dalam praktiknya, programmer menggunakan beragam bahasa pemrograman untuk menghasilkan perangkat lunak sesuai kebutuhan. Beberapa bahasa yang paling populer dan banyak digunakan saat ini antara lain Python, yang terkenal karena sintaksnya sederhana serta banyak dipakai untuk kecerdasan buatan, analisis data, dan pengembangan web, dengan contoh aplikasi populer seperti Instagram dan Spotify. Java digunakan luas dalam pengembangan aplikasi Android dan sistem berbasis enterprise, misalnya aplikasi perbankan mobile dan platform e-commerce besar seperti LinkedIn. C++ dikenal dengan performa tinggi sehingga banyak digunakan dalam pengembangan game populer seperti Counter-Strike, mesin game Unreal Engine, hingga perangkat lunak grafis seperti Adobe Photoshop. JavaScript menjadi tulang punggung aplikasi web interaktif, sementara variannya yaitu TypeScript semakin populer karena menambahkan fitur pengetikan statis yang memudahkan pengembangan aplikasi berskala besar, contohnya pada Microsoft Teams dan Slack.

Selain itu, Swift menjadi bahasa utama dalam pengembangan aplikasi untuk perangkat Apple seperti iOS, macOS, dan watchOS, dengan contoh aplikasi seperti Twitter dan Airbnb versi iOS. Kotlin juga semakin banyak digunakan sebagai bahasa resmi pengembangan Android yang modern dan efisien, menggantikan dominasi Java di platform tersebut, dengan contoh aplikasi seperti Pinterest dan Trello. Dengan penguasaan bahasa-bahasa pemrograman populer ini beserta penerapannya pada aplikasi nyata, programmer dapat lebih fleksibel dalam menghadapi berbagai tantangan teknologi dan meningkatkan daya saing dalam industri digital global.

Tabel 1. 1 Contoh Penggunaaan Bahasa Pemrograman

Tingkatan	Bahasa Mesin	Bahasa Tingkat Rendah	Bahasa Tingkat Tinggi
Tahun Muncul	1940-an	1950-an	1960-an – sekarang
Karakteristik	Instruksi berupa kode biner (0 dan 1), langsung dijalankan oleh prosesor	Menggunakan mnemonik singkat (MOV, ADD, SUB), dekat dengan arsitektur prosesor	Sintaks menyerupai bahasa manusia, abstraksi lebih tinggi, portabel
Kelebihan	Eksekusi sangat cepat, langsung dipahami komputer	Lebih mudah daripada biner, kontrol langsung ke hardware, efisien	Mudah dipahami, cepat dipelajari, dapat dijalankan di berbagai platform
Kekurangan	Sangat sulit dipahami manusia, rawan kesalahan, tidak portabel	Tetap sulit untuk aplikasi besar, butuh pemahaman detail hardware	Performa sedikit lebih lambat dibanding assembly atau biner
Contoh Bahasa	– (langsung biner)	Assembly	Python, Java, C, C++, PHP, JS, TypeScript, Swift, Kotlin
Contoh Aplikasi	Firmware sederhana, instruksi prosesor awal	Sistem operasi lama, driver perangkat keras	Aplikasi web, mobile, AI, game, sistem industri

Selain manfaat praktis, pemrograman juga berperan penting dalam membentuk pola pikir logis dan kemampuan problem-solving, terutama bagi generasi muda. Belajar pemrograman melatih seseorang untuk berpikir sistematis, mengenali pola, dan menyusun solusi secara terstruktur. Oleh karena itu, banyak negara mulai memasukkan pemrograman ke dalam kurikulum sekolah sebagai keterampilan abad ke-21. Dengan kemampuan pemrograman, seseorang tidak hanya menjadi pengguna teknologi, tetapi juga dapat menjadi pencipta dan pengembang solusi digital yang membawa manfaat luas bagi masyarakat.

Gambar 1. 1 Peran Pemrograman dan Kehidupan

Pemrograman memiliki peran yang sangat penting dalam kehidupan modern karena menjadi dasar dari berbagai teknologi yang kita gunakan setiap hari. Dalam bidang aplikasi mobile, pemrograman memungkinkan hadirnya layanan chatting, transportasi online, belanja digital, hingga pembayaran cashless yang membuat aktivitas sehari-hari lebih praktis. Di sektor kesehatan, teknologi pemrograman digunakan untuk mengelola rekam medis digital, menciptakan aplikasi kesehatan, serta mendukung diagnosa berbasis kecerdasan buatan yang membantu dokter dalam memberikan perawatan lebih tepat dan cepat.

Tidak hanya itu, di dunia pendidikan, pemrograman mendukung berkembangnya platform e-learning, aplikasi belajar interaktif, serta kelas online yang membuat proses belajar lebih fleksibel, menarik, dan dapat diakses dari mana saja. Pada sektor keuangan, pemrograman menghadirkan inovasi seperti mobile banking, dompet digital (e-wallet),

dan sistem keamanan transaksi yang menjaga kenyamanan serta keamanan pengguna. Sementara itu, dalam bidang otomasi dan kecerdasan buatan, pemrograman berperan dalam menciptakan robot industri, smart home, hingga asisten digital seperti Alexa dan Google Assistant yang semakin mempermudah aktivitas manusia.

Secara keseluruhan, teknologi yang lahir dari pemrograman tidak hanya mendukung dunia digital, tetapi juga melekat erat dalam kehidupan sehari-hari. Pemrograman menjadikan manusia bukan hanya pengguna teknologi, melainkan juga pencipta solusi defu yang mampu meningkatkan kualitas hidup dan menjawab tantangan zaman.

BAB 2
DASAR LOGIKA PEMROGRAMAN

A. Pendahuluan

Logika pemrograman adalah dasar keterampilan yang harus dikuasai sebelum seseorang menulis program menggunakan bahasa pemrograman tertentu (Avorizano et al., 2025). Dengan memahami logika pemrograman, seorang programmer dapat menyusun langkah-langkah penyelesaian masalah secara runtut, terstruktur, dan efisien. Pada bab ini akan dibahas beberapa konsep utama, yaitu algoritma, flowchart, pseudocode, serta jenis-jenis algoritma dan penerapannya.

B. Algoritma

1. Definisi Algoritma

Algoritma adalah serangkaian langkah-langkah logis, sistematis, dan terstruktur yang disusun untuk menyelesaikan suatu permasalahan atau mencapai tujuan tertentu (Cahyaningtyas & Gudiato, 2025). Dalam dunia komputasi, algoritma menjadi dasar dari setiap proses pemrograman karena menentukan bagaimana sebuah program akan bekerja untuk mengolah input menjadi output yang sesuai. Penyusunan algoritma biasanya diawali dengan memahami masalah secara menyeluruh, kemudian memecahnya menjadi bagian-bagian kecil yang lebih sederhana, sehingga dapat dirumuskan dalam bentuk langkah-langkah logis yang mudah diimplementasikan menggunakan bahasa pemrograman.

Secara umum, algoritma dapat didefinisikan sebagai serangkaian instruksi yang jelas, terstruktur, dan memiliki tujuan akhir tertentu (Panggabean et al., 2021). Algoritma selalu melibatkan tiga komponen utama, yaitu input (data masukan yang akan diproses), proses

(serangkaian instruksi untuk mengolah input), dan output (hasil yang diharapkan). Untuk memudahkan implementasi, algoritma biasanya digamdefkan dalam bentuk pseudocode atau flowchart sebelum dituangkan ke dalam kode program.

Peran algoritma sangat penting dalam pemrograman karena menjadi pondasi yang memastikan sebuah program dapat bekerja secara efektif. Algoritma yang baik mampu meningkatkan efisiensi, baik dari sisi waktu (time complexity) maupun sumber daya (space complexity). Dengan algoritma yang tepat, suatu masalah dapat diselesaikan lebih cepat, lebih hemat memori, dan hasilnya lebih optimal. Efisiensi inilah yang membuat algoritma menjadi inti dari banyak inovasi teknologi modern.

Contoh penerapan algoritma dapat ditemukan dalam berbagai aspek kehidupan. Mesin pencari internet menggunakan algoritma pencarian dan perankingan untuk menampilkan informasi yang relevan. Aplikasi peta digital mengandalkan algoritma shortest path untuk menentukan rute tercepat. Di bidang keuangan, algoritma digunakan untuk mendeteksi penipuan transaksi. Bahkan dalam kecerdasan buatan dan big data, algoritma berperan dalam memproses data dalam jumlah besar untuk menghasilkan prediksi, rekomendasi, maupun keputusan otomatis. Hal ini menunjukkan bahwa algoritma bukan hanya konsep akademis, tetapi solusi nyata yang menghubungkan teknologi dengan kebutuhan manusia sehari-hari.

2. Karakteristik Algoritma yang Baik

a. Jelas

Setiap langkah dalam algoritma harus ditulis menggunakan bahasa yang sederhana dan mudah dipahami. Tujuannya adalah agar alur kerja dapat diikuti dengan jelas tanpa menimbulkan kebingungan, baik saat dibaca manusia maupun ketika diterjemahkan ke dalam kode program. Oleh karena itu, setiap instruksi harus dirumuskan secara langsung dan tidak bertele-tele. Selain itu, algoritma tidak boleh mengandung makna ganda atau ambigu. Setiap langkah harus dijelaskan secara spesifik agar tidak menimbulkan interpretasi yang berbeda. Misalnya, daripada menuliskan "hitung nilai rata-rata",

algoritma harus memerinci langkah-langkahnya: "jumlahkan semua nilai, kemudian bagi dengan banyaknya data". Dengan penjelasan yang jelas seperti ini, proses pengubahan dari algoritma ke program dapat berjalan lebih tepat dan konsisten.

b. Sistematis

Algoritma harus tersusun secara berurutan dari langkah awal hingga langkah akhir dengan logika yang jelas. Setiap langkah disusun agar saling berkaitan dan mendukung penyelesaian masalah, sehingga alurnya dapat diikuti tanpa menimbulkan kebingungan. Urutan yang tepat juga membantu memastikan bahwa setiap proses dijalankan pada waktu yang benar, karena langkah yang ditempatkan tidak pada posisinya dapat menyebabkan hasil yang keliru. Penyusunan yang sistematis tidak hanya membantu pembaca memahami proses pemecahan masalah, tetapi juga memudahkan programmer ketika menerjemahkannya ke dalam kode program.

Dengan alur yang runtut, kesalahan dalam algoritma juga lebih mudah dilacak apabila terjadi kekeliruan pada salah satu tahap. Programmer dapat menelusuri setiap langkah secara berurutan untuk menemukan titik masalah tanpa harus memeriksa seluruh proses secara acak. Jika struktur langkah tersusun dengan baik, bagian yang salah dapat diidentifikasi dan diperbaiki dengan lebih cepat. Hal ini menjadikan algoritma lebih efisien, terkontrol, dan siap diimplementasikan secara akurat ke dalam program komputer.

c. Efisien

Algoritma yang baik mampu menyelesaikan permasalahan dengan cepat dan tepat. Efisiensi dapat dilihat dari dua sisi, yaitu waktu (time complexity) dan sumber daya yang digunakan (space complexity). Semakin sedikit langkah yang dibutuhkan dan semakin hemat penggunaan memori, maka algoritma tersebut dianggap lebih efisien.

Selain itu, algoritma yang efisien juga membantu meningkatkan performa program, terutama ketika bekerja dengan data yang besar. Dengan memilih pendekatan yang tepat, proses komputasi dapat berjalan lebih ringan sehingga program dapat merespons lebih cepat dan tidak membebani sistem.

d. Pasti Berakhir (Terminated)

Algoritma harus memiliki titik akhir yang jelas. Artinya, setelah mengikuti urutan langkah-langkahnya, algoritma pasti akan berhenti dengan menghasilkan output yang diinginkan. Jika algoritma tidak memiliki terminasi, maka akan berpotensi menjadi infinite loop atau perulangan tak berujung, yang membuat solusi tidak pernah selesai.

3. Contoh Algoritma

Algoritma Menentukan Bilangan Terbesar dari Dua Angka
Masalah: Diberikan dua bilangan, tentukan bilangan mana yang lebih besar.
Langkah-langkah:

1) Mulai
2) Masukkan nilai A dan B
3) Jika A > B maka tampilkan "A lebih besar"
4) Jika tidak maka tampilkan "B lebih besar"
5) Selesai

C. Pseudocode

Pseudocode adalah cara menuliskan algoritma secara tertulis menggunakan gaya bahasa yang menyerupai bahasa pemrograman namun tetap mudah dipahami oleh Manusia (Sulianta, 2025). Tidak seperti bahasa pemrograman asli, pseudocode tidak terikat oleh aturan sintaks yang ketat. Hal ini membuat penulis dapat fokus pada logika penyelesaian masalah, alur proses, percabangan keputusan, serta perulangan tanpa harus memikirkan aturan penulisan yang rumit.

C++ :

```
#include <iostream>
using namespace std;

int main() {
    int A, B;

    cout << "Masukkan A: ";
    cin >> A;

    cout << "Masukkan B: ";
    cin >> B;

    if (A > B) {
        cout << "A lebih besar" << endl;
    } else {
        cout << "B lebih besar atau sama dengan A" <<
endl;
    }

    return 0;
}
```

Java :

```
import java.util.Scanner;

public class Perbandingan {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan A: ");
        int A = input.nextInt();

        System.out.print("Masukkan B: ");
        int B = input.nextInt();
```

```
        if (A > B) {
            System.out.println("A lebih besar");
        } else {
            System.out.println("B  lebih  besar  atau
sama dengan A");
        }
    }
}
```

PHP:

```
<?php
$A = 10;
$B = 20;

if ($A > $B) {
    echo "A lebih besar";
} else {
    echo "B lebih besar atau sama dengan A";
}
?>
```

F. Instalasi Phyton

Python merupakan bahasa pemrograman populer yang digunakan dalam berbagai bidang seperti pengembangan perangkat lunak, analisis data, kecerdasan buatan, machine learning, hingga automasi tugas. Untuk dapat menjalankan Python, diperlukan beberapa langkah instalasi dan pemilihan alat (tools) yang sesuai dengan kebutuhan pengguna.

Python dapat dijalankan melalui berbagai alat, mulai dari instalasi standar menggunakan Python Installer, distribusi Anaconda, editor profesional seperti Visual Studio Code, hingga platform cloud seperti Google Colab. Pemilihan alat terbaik tergantung pada kebutuhan pengguna, spesifikasi perangkat, dan jenis proyek yang dikerjakan. Dengan

Instalasi Anaconda:

a. Unduh installer dari: *https://www.anaconda.com/download*
b. Pilih sistem operasi yang digunakan.
c. Jalankan installer dan ikuti instruksi.
d. Setelah selesai, dapat dijalankan dengan: Anaconda Prompt / Jupyter Notebook / Anaconda Navigator.

3. Visual Studio Code (VS Code)

Visual Studio Code merupakan editor kode modern dan ringan yang banyak digunakan untuk menulis program Python (Aldi et al., 2025). Fungsi VS Code adalah menulis, mengedit, dan menjalankan kode Python, mendukung debugging, git version control, ekstensi dan mendukung notebook interaktif (Jupyter) melalui extension Python.

Integrasi VS Code dengan Python

VS Code tidak menyertakan Python secara bawaan. Pengguna perlu terlebih dahulu menginstal Python melalui:

- Installer resmi Python, atau
- Anaconda.

Setelah Python terinstal, VS Code dapat mendeteksi interpreter melalui menu: *Command Palette → Python: Select Interpreter*

4. Google Colab

Google Colaboratory (Colab) adalah platform berbasis cloud yang memungkinkan pengguna menjalankan Python tanpa instalasi di komputer (Octarina et al., 2025). Google Colab sangat cocok bagi pengguna dengan spesifikasi komputer terbatas atau untuk pembelajaran berbasis notebook. Keunggulan menggunakan Google Colab adalah tidak membutuhkan instalasi Python, dapat dijalankan di browser menggunakan akun Google, mendukung pustaka data science secara default (pandas, numpy, tensorflow) dan menyediakan GPU/TPU gratis.

G. Instalasi C++

Dalam mempelajari bahasa pemrograman C++, kita membutuhkan perangkat lunak yang dapat digunakan untuk menulis, menjalankan, dan melakukan debugging terhadap program. Di Windows, terdapat beberapa pilihan alat yang dapat digunakan, antara lain Code::Blocks, Dev-C++, Visual Studio Code, dan compiler MinGW. Setiap alat memiliki kelebihannya masing-masing, sehingga pengguna dapat memilih yang paling sesuai dengan kebutuhan pembelajaran atau pengembangan proyek.

Pada bagian ini akan dijelaskan langkah-langkah instalasi keempat alat tersebut secara lengkap sebagai panduan bagi mahasiswa maupun pemula dalam memulai pemrograman C++.

1. Instalasi Code::Blocks

Code::Blocks adalah salah satu IDE yang populer untuk pemrograman C dan C++. IDE ini bersifat ringan, mudah digunakan, dan biasanya direkomendasikan untuk pemula karena tampilannya sederhana namun lengkap dengan fitur debugging.

Langkah Instalasi Code::Blocks :

a. Buka situs resmi Code::Blocks di *https://www.codeblocks.org/downloads/binaries/*
b. Pilih installer dengan nama: *codeblocks-XX.XXmingw-setup.exe* (versi ini sudah termasuk compiler MinGW)
c. Jalankan file installer dan klik Next serta I Agree.
d. Pilih Full Installation, lalu klik Install.
e. Tunggu proses instalasi selesai dan klik Finish.
f. Setelah terbuka, masuk ke menu Settings → Compiler, dan pastikan compiler yang muncul adalah GNU GCC Compiler.

Dengan demikian, Code::Blocks sudah siap digunakan tanpa perlu memasang compiler tambahan.

2. Java Runtime Environment (JRE)

JRE adalah komponen untuk menjalankan aplikasi Java. Berbeda dengan JDK, JRE tidak dapat melakukan kompilasi karena tidak memiliki javac. Biasanya digunakan oleh pengguna aplikasi, bukan programmer.

Cara Instalasi JRE

a. Unduh dari situs Oracle.
b. Instal seperti aplikasi biasa.
c. Setelah instalasi, program Java dapat dijalankan, namun pengembangan tetap memerlukan JDK.

3. Integrated Development Environment (IDE) untuk Java

IDE adalah perangkat lunak yang menyediakan fasilitas lengkap untuk menulis kode, mengelola proyek, mendeteksi error, hingga menjalankan program secara langsung.

Berikut IDE populer untuk Java:

a. BlueJ

BlueJ dirancang khusus untuk pembelajaran dan pengajaran OOP. Tampilan sederhana membuatnya ideal untuk siswa atau pemula.

Cara Instalasi BlueJ :

- Instal JDK terlebih dahulu.
- Buka situs resmi BlueJ.
- Unduh installer sesuai sistem operasi.
- Jalankan installer dan selesaikan proses instalasi.
- BlueJ akan otomatis mendeteksi JDK yang terpasang.

b. NetBeans

NetBeans adalah IDE lengkap yang dulu menjadi IDE resmi dari Oracle. Mendukung Java, PHP, C++, dan berbagai bahasa lain.

Cara Instalasi NetBeans :

- Unduh NetBeans dari situs Apache NetBeans.
- Pastikan JDK sudah terpasang.
- Jalankan installer dan pilih paket Java SE atau Java Full.
- Ikuti langkah-langkah instalasi hingga selesai.

4. Editor Tambahan untuk Java

a. JGrasp

IDE ringan yang menyediakan visualisasi struktur data. Cocok untuk pembelajaran array, tree, dan graph.

Instalasi :

- Unduh installer dari situs jGRASP.
- Instal JDK terlebih dahulu.
- Jalankan installer dan buka proyek defu.

b. DrJava

IDE paling sederhana dan ringan, sering digunakan untuk pemula atau komputer berspesifikasi rendah.

Instalasi :

- Unduh file .jar DrJava.
- Jalankan melalui double-click karena DrJava berbasis Java.

c. Android Studio

Meskipun khusus untuk pengembangan Android, Android Studio mendukung Java sebagai bahasa utama.

Instalasi :

- Unduh dari situs Android Developers.
- Instal JDK terlebih dahulu.
- Jalankan installer dan ikuti konfigurasi SDK.

d. Compiler dan Editor Online untuk Java

Compiler online sering digunakan untuk pembelajaran cepat tanpa instalasi. Contoh alat: Replit / Jdoodle / OnlineGDB/ NetBeansOnline Editor.

Cara Menggunakan:

- Buka situs pilihan.
- Pilih bahasa Java.
- Masukkan kode ke editor.
- Klik tombol "Run".

Tidak memerlukan instalasi JDK karena sudah tersedia di server.

e. Build Tools: Maven dan Gradle

Dalam proyek profesional, Java membutuhkan alat otomatisasi untuk mengelola dependensi dan proses build.

1) Maven : Menggunakan file pom.xml untuk mengatur pustaka dan konfigurasi project.
2) Gradle : Lebih modern dan lebih cepat, menggunakan file build.gradle.

Cara Instalasi Maven/Gradle

- Instal JDK.
- Unduh Maven/Gradle ZIP dari situs resmi.
- Ekstrak dan tambahkan path ke Environment Variables.
- Verifikasi dengan perintah:
- mvn -v
- gradle -v

Untuk menjalankan Java, alat paling utama adalah JDK. Semua IDE dan editor lainnya seperti BlueJ, NetBeans, IntelliJ, Eclipse, dan VS Code tetap membutuhkan JDK agar kode dapat dikompilasi. Selain itu ada alternatif seperti JGrasp, DrJava, Replit, dan compiler online lain yang cocok untuk pembelajaran dan eksperimen cepat. Alat-alat ini memberikan fleksibilitas kepada pengguna, baik yang ingin belajar dasar Java dengan tampilan sederhana, maupun yang ingin membangun aplikasi besar dengan IDE profesional.

I. Instalasi PHP

PHP adalah bahasa pemrograman server-side yang digunakan untuk mengembangkan halaman web dinamis. Agar PHP dapat dijalankan, dibutuhkan serangkaian perangkat yang berbeda dari bahasa pemrograman lain seperti C++ atau Java. PHP tidak dijalankan langsung oleh browser ataupun sistem operasi, melainkan oleh web server. Bab ini membahas alat-alat yang dibutuhkan untuk menjalankan PHP beserta langkah instalasinya.

- Letakkan file PHP di folder: *C:\xampp\htdocs*
- Jalankan melalui browser: *http://localhost/namafile.php*

b. WAMP Server

Alternatif khusus Windows dengan antarmuka lebih ringan.
Komponen:Windows, Apache, MySQL/MariaDB dan PHP.
Instalasi WAMP:

- Unduh dari situs WAMP.
- Instal dan jalankan server.
- Simpan file PHP di folder www.
- Akses melalui *http://localhost*.

c. Laragon

Sangat cepat dan ringan, banyak digunakan developer Laravel dan PHP modern. Fitur: Auto virtual host, Mendukung Apache & Nginx, MySQL/MariaDB, phpMyAdmin, HeidiSQL dan Cocok untuk project modern

Instalasi Laragon:

- Unduh Laragon.
- Install dan jalankan.
- Letakkan project PHP di folder /www.

d. AMPPS

Alternatif XAMPP/WAMP dengan lebih banyak modul.
Instalasi

- Unduh dari situs AMPPS.
- Instal seperti biasa.
- Jalankan Apache dan MySQL.

6. PHP CLI (Command Line Interface)

Selain melalui web server, PHP dapat dijalankan langsung lewat terminal menggunakan CLI. Perintah menjalankan file PHP: *php namafile.php.* PHP CLI sangat penting untuk aplikasi berbasis framework modern seperti Laravel yang memerlukan perintah Artisan.

J. Struktur Dasar Kode Program

1. Struktur Dasar Penulisan Kode Program Python

Python dikenal sebagai salah satu bahasa pemrograman yang memiliki tata penulisan sederhana dan mudah dipahami. Hampir semua konsep dasarnya dirancang agar dapat dipelajari oleh pemula, namun tetap cukup kuat digunakan dalam proyek profesional. Supaya dapat menulis program Python dengan benar, penting untuk memahami bagaimana struktur dasar sebuah kode Python dibentuk.

1) **Tidak Memerlukan Deklarasi Class atau Main (Namun Bisa Digunakan)**

   Berbeda dari Java atau C++, Python tidak mengharuskan program diawali dengan deklarasi class ataupun fungsi main. Kita bisa langsung menuliskan perintah-perintah sederhana pada defis pertama file Python dan program sudah bisa dijalankan.

   Contoh program Python yang sangat sederhana:

   ```
   print("Halo, selamat datang di dunia Python!")
   ```

   Walaupun demikian, Python tetap mendukung pembuatan class serta fungsi main() apabila dibutuhkan dalam program yang lebih kompleks. Namun untuk pemula, cukup menuliskan instruksi secara langsung di dalam file .py.

2) **Menggunakan Indentasi Sebagai Penanda Blok Kode**

   Salah satu ciri khas Python adalah penggunaan indentasi (spasi atau tab) untuk menentukan blok kode. Python tidak menggunakan tanda kurung kurawal seperti pada Java atau

(untuk menampilkan teks) dan **`cin`** (untuk mengambil input pengguna).

Karena C++ menerapkan konsep modular, fitur-fitur bahasa tidak digabung dalam satu paket besar, melainkan dipisah ke dalam banyak header. Contohnya:

- `#include <cmath>` untuk operasi matematika,
- `#include <fstream>` untuk memproses file,
- `#include <string>` untuk manipulasi teks.

Dengan model ini, program menjadi lebih efisien karena hanya memuat fungsi yang benar-benar dipakai. Namun konsekuensinya, programmer wajib menyertakan header yang dibutuhkan sebelum menggunakan suatu fitur.

3) **Fungsi main() Sebagai Titik Eksekusi Program**

Bagian:

```
int main() { }
```

adalah fungsi utama tempat eksekusi program dimulai. Hampir seluruh aplikasi C++ memiliki fungsi ini, karena sistem operasi akan memulai jalannya program melalui fungsi `main()`.

Isi dari fungsi ditulis di antara tanda kurung kurawal `{ }`. Di sinilah seluruh instruksi utama diletakkan.

Kata kunci int di depan main menunjukkan bahwa fungsi ini harus menghasilkan sebuah nilai bertipe integer sebagai tanda status program setelah selesai dijalankan. Nilai tersebut dikembalikan melalui perintah return.

4) **Menampilkan Output Menggunakan std::cout**

Untuk menampilkan teks ke layar, C++ menyediakan objek cout, yang merupakan bagian dari namespace standar (`std`).

Contoh penulisan:

```
std::cout << "Hello World!";
```

Tanda << digunakan untuk "mengalirkan" data ke layar. Setiap instruksi harus diakhiri tanda titik koma `;`, yang menjadi aturan wajib dalam bahasa C++.

`std::cout` sebenarnya terdiri dari dua bagian:

- `std` → nama namespace tempat cout berada
- `cout` → objek untuk menampilkan teks

Namespace berfungsi untuk mencegah bentrokan nama (name collision). Jika ada dua fungsi dengan nama sama tetapi berada pada namespace berbeda, keduanya tetap bisa digunakan.

Untuk menampilkan lebih dari satu teks, kita dapat memanggil `cout` beberapa kali:

```
std::cout << "Hello World!\n";
std::cout << "Sedang mempelajari bahasa C++ ";
std::cout << "pada pertemuan ini.";
```

Karakter `\n` digunakan untuk berpindah defis.

5) **Menggunakan using namespace std**

Agar tidak perlu menulis `std::` berulang kali, kita dapat menuliskan:

```
using namespace std;
```

Dengan demikian, perintah:

```
cout << "Hello World!";
```

sudah cukup tanpa menuliskan `std::cout`.

Secara umum, kedua pendekatan sama validnya. Penulisan dengan `using namespace std` lebih sering digunakan dalam pembelajaran dasar karena lebih ringkas, sedangkan penulisan

dengan `std::` lebih disarankan pada proyek besar untuk menghindari konflik nama.

6) **Perintah return 0; sebagai Penanda Akhir Program**
Instruksi:

```
return 0;
```

adalah tanda bahwa fungsi `main()` telah selesai dijalankan. Nilai 0 menunjukkan program berakhir dengan sukses (*exit success*). Nilai selain 0 biasanya menandakan ada kesalahan atau kondisi tertentu yang ingin disampaikan ke sistem operasi. Sebagian compiler modern akan secara otomatis menambahkan `return 0` jika tidak ditulis. Namun, dalam penulisan program yang baik dan terstruktur, menuliskannya secara eksplisit tetap direkomendasikan.

Struktur dasar C++ terdiri dari header, fungsi `main()`, mekanisme output dengan `cout`, serta penggunaan namespace untuk mengelompokkan fungsi. Walaupun sederhana, pemahaman pada bagian ini sangat penting karena menjadi dasar dari berbagai konsep lanjutan seperti fungsi, kelas, file handling, hingga OOP. Dengan menguasai kerangkanya sejak awal, pembelajaran C++ akan menjadi lebih mudah dan sistematis.

3. Struktur Dasar Penulisan Kode Program Java

Dalam pemrograman Java, setiap program memiliki pola penulisan tertentu yang wajib diikuti agar kode dapat dijalankan dengan benar. Walaupun contoh program yang digunakan dalam pembelajaran awal terlihat sederhana, sebenarnya contoh tersebut sudah memperlihatkan bagaimana kerangka dasar dari sebuah program Java dibentuk.

Untuk sementara, Anda cukup memahami bahwa *blok main* adalah tempat utama untuk menulis logika program. Seperti class, blok main juga dibatasi oleh kurung kurawal:

```
public static void main(String args[]){
    ...
}
```

Semua kode di dalamnya akan dieksekusi secara berurutan ketika program dijalankan.

c. Menggunakan System.out.println() untuk Menampilkan Output

Java menyediakan perintah standar untuk menampilkan teks ke layar, yaitu:

```
System.out.println();
```

Metode ini digunakan untuk mengeluarkan tulisan atau informasi yang ingin ditampilkan kepada pengguna. Pesan yang akan ditampilkan ditulis di dalam tanda kurung dan diberi tanda kutip:

```
System.out.println("Belajar Java itu menyenangkan!");
```

Anda dapat memanggil perintah ini sebanyak yang dibutuhkan:

```
System.out.println("Stt... sedang serius belajar bahasa Java");
System.out.println("di Duniailkom");
System.out.println("Semangaaat...");
```

Setiap pemanggilan akan membuat teks tampil di defis defu.

4. Struktur Dasar Penulisan Kode Program PHP

PHP merupakan bahasa pemrograman yang dirancang khusus untuk membangun halaman web dinamis. Tidak seperti beberapa bahasa lain yang membutuhkan struktur kompleks sejak awal, PHP memiliki bentuk dasar yang sangat fleksibel. Meski demikian, ada pola umum yang perlu dipahami agar penulisan kode PHP menjadi rapi, konsisten, dan terbaca dengan baik.

1. Tag Pembuka dan Penutup PHP

   Setiap kode PHP harus berada di dalam *tag* khusus agar dapat dieksekusi oleh server. Format standarnya adalah:

   ```
   <?php
      // kode PHP ditulis di sini
   ?>
   ```

   Browser hanya akan mengenali kode yang ditulis di dalam tag tersebut sebagai instruksi PHP. Di luar tag, sebuah file tetap dapat memuat HTML, CSS, atau teks lainnya sehingga PHP dapat bercampur langsung dengan struktur halaman web.

   Contoh PHP di dalam HTML:

   ```
   <html>
   <body>
       <h1>Selamat datang!</h1>
       <?php
           echo "Ini teks dari PHP";
       ?>
   </body>
   </html>
   ```

2. Perintah Output Menggunakan echo atau print

   Untuk menampilkan teks di halaman web, PHP menyediakan perintah echo dan print.

   Contoh:

   ```
   echo "Halo, PHP!";
   ```

Perintah ini digunakan sangat sering, terutama saat ingin memunculkan hasil perhitungan, teks, nilai variabel, ataupun debugging.
Kedua perintah ini hampir sama fungsinya, tetapi `echo` lebih umum dipakai karena sedikit lebih cepat.

3. Wajib Menggunakan Tanda Titik Koma
   PHP mewajibkan setiap perintah diakhiri dengan tanda ; (semicolon). Tanpa tanda ini, program akan menghasilkan error.
   Contoh benar:

   ```
   echo "Selamat belajar PHP!";
   ```

   Contoh salah:

   ```
   echo "Selamat belajar PHP!"
   ```

   Kesalahan tanda titik koma merupakan salah satu error paling sering ditemui pemula.

4. Variabel Diawali dengan Simbol $
   PHP menggunakan tanda dolar $ untuk mendefinisikan variabel. Nama variabel tidak memerlukan deklarasi tipe data, karena PHP menggunakan sistem *dynamic typing*.
   Contoh:

   ```
   $nama = "Andi";
   $umur = 20;
   $tinggi = 165.5;
   ```

   Variabel dapat berisi angka, teks, array, dan berbagai struktur data lain.

5. Fungsi (Function)

   Untuk menyusun program yang lebih terstruktur, PHP memungkinkan pembuatan fungsi.

   Contoh:

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

sapa("Budi");
```

   Fungsi sangat berguna saat ingin menghindari penulangan kode.

K. Aturan Dasar Kode Program

1. Aturan Dasar Penulisan Kode Program Phyton

Setelah Python berhasil dipasang dan interpreter dapat dijalankan dengan baik, langkah berikutnya adalah memahami aturan dasar penulisan sintaks yang menjadi fondasi saat membuat program. Python memiliki karakteristik unik yang membedakannya dari banyak bahasa pemrograman lain, dan pemahaman terhadap aturan–aturan ini akan memudahkan proses belajar maupun penulisan kode.

1) **Pemisah Perintah (Statement Terminator)**

   Pada banyak bahasa pemrograman lain, setiap perintah biasanya harus diakhiri dengan tanda titik koma ;. Dalam Python, aturan ini tidak berlaku. Pemisahan perintah dilakukan secara otomatis oleh karakter newline—yaitu perpindahan defis ketika kita menekan tombol Enter.

   Contoh:

```
print("Hello World")
web = "Python"
print("Belajar bahasa " + web)
print("Semangat belajar!")
```

Setiap defis dianggap sebagai satu instruksi berbeda selama berada pada defis yang terpisah.

Python sebenarnya mengizinkan penggunaan titik koma, seperti:

```
print("Hello World");
web = "Python";
```

Namun tanda tersebut hanya diperlukan jika kita ingin menempatkan beberapa perintah dalam satu defis:

```
print("Hello"); print("Belajar Python")
```

Meskipun memungkinkan, format seperti ini tidak dianjurkan karena membuat kode sulit dibaca. Pada praktik yang baik, setiap perintah sebaiknya tetap ditulis pada defis terpisah.

2) **Sensitivitas Huruf Besar dan Kecil (Case Sensitivity)**
Sama seperti C++, Python juga merupakan bahasa yang case sensitive. Artinya, huruf besar dan huruf kecil diperlakukan sebagai hal yang berbeda.
Contoh:
- `print` → benar
- `Print` → salah
- `PRINT` → salah

Hal yang sama berlaku untuk variabel maupun nama fungsi. Misalnya, jumlah, Jumlah, dan JUMLAH semuanya dianggap sebagai nama variabel yang berbeda. Jika kita menulis perintah `Print()` atau `PRINT()`, Python akan menghasilkan pesan kesalahan karena interpreter tidak mengenali perintah tersebut.

3) **Indentasi (Indentation)**
Salah satu ciri khas Python adalah penggunaan indentasi sebagai penanda blok kode. Jika pada bahasa lain blok dimulai dengan { }, Python menentukan batas blok dengan cara memberi jarak (spasi atau tab) di awal defis. Artinya, indentasi bukan sekadar pemanis visual, tetapi bagian penting dari sintaks.

5) **Identifier dalam C++**

Identifier adalah nama yang digunakan untuk menandai elemen dalam program—misalnya variabel, konstanta, fungsi, atau objek lainnya.

Contoh:

```
int total;
double harga_defang;
```

Pada contoh tersebut, `total` dan `harga_defang` merupakan identifier.

Beberapa aturan penulisan identifier:

1) Tidak boleh menggunakan keyword.
2) Hanya boleh terdiri atas huruf, angka, dan garis bawah _.
3) Karakter pertama harus huruf atau _, tidak boleh angka.
4) Disarankan tidak memulai identifier dengan _ untuk menghindari konflik dengan sistem.
5) Beberapa compiler membatasi panjang identifier, misalnya 31 karakter.

Dengan mengikuti aturan ini, kita dapat menuliskan nama variabel dan fungsi dengan lebih rapi, mudah dipahami, dan sesuai standar pemrograman C++. Pemahaman dasar mengenai case sensitivity, komentar, statement, keyword, dan identifier sangat penting sebelum mempelajari konsep-konsep C++ yang lebih kompleks. Aturan-aturan ini menjadi fondasi bagi setiap program C++, baik sederhana maupun berskala besar. Dengan membiasakan penulisan kode yang benar sejak awal, proses belajar dan debugging akan menjadi jauh lebih mudah.

3. Aturan Dasar Penulisan Kode Program Java

Dalam mempelajari bahasa pemrograman Java, memahami aturan dasar penulisan kode merupakan fondasi yang sangat penting. Aturan ini menentukan bagaimana compiler membaca instruksi, membedakan struktur program, dan menafsirkan perintah yang kita tulis. Beberapa aspek mendasar yang perlu diperhatikan meliputi sensitivitas huruf (case sensitivity), penggunaan whitespace, komentar, statement, serta penamaan identifier.

1) **Sensitivitas Huruf Kapital (Case Sensitivity)**

Java adalah bahasa yang membedakan huruf besar dan huruf kecil. Artinya, penulisan `System`, `system`, dan `SYSTEM` dianggap sebagai tiga hal yang berbeda. Kesalahan kecil dalam penggunaan huruf dapat menyebabkan error.

Misalnya:

```
class HelloBelajar {
    public static void main(String args[]){
        system.out.println("Belajar Java");
    }
}
```

Kode di atas akan menghasilkan error karena Java hanya mengenali System, bukan system. Sensitivitas huruf ini berlaku pada semua elemen Java, termasuk nama variabel, metode, kelas, dan keyword tertentu.

2) **Penulisan Komentar dalam Java**

Komentar digunakan untuk memberikan catatan, penjelasan, atau instruksi tambahan pada kode. Sesuai fungsinya, komentar tidak dijalankan oleh program dan hanya dibaca oleh programmer.

Java menyediakan tiga jenis komentar:

1) Single Line Comment

Ditulis dengan // pada awal defis:

```
// Menampilkan teks ke layar
System.out.println("Belajar Java");
```

2) Multi Line Comment

Digunakan untuk komentar lebih dari satu defis:

```
/*
   Contoh komentar beberapa defis
   Penjelasan dapat ditulis panjang di sini
*/
```

3) Documentation Comment (Javadoc)
   Jenis komentar ini dibuat khusus untuk menghasilkan dokumentasi otomatis menggunakan Javadoc:

```
/**
 * Menampilkan teks ke layar
 * Digunakan sebagai bagian dari dokumentasi
program
 */
```

Komentar juga sering dipakai untuk sementara menonaktifkan defis kode saat proses debugging tanpa harus menghapusnya.

3) **Statement dalam Java**
   Statement adalah satu unit instruksi yang dibaca dan dijalankan oleh Java. Sebagian besar statement diakhiri dengan tanda titik koma (;).
   Contoh:

```
int nilai = 80;
hasil = a + b;
System.out.println("Belajar Java");
```

Beberapa statement dapat terdiri dari beberapa defis, misalnya struktur *control flow* seperti `if`, `for`, atau `while`. Statement yang berisi lebih dari satu defis biasanya berada dalam blok `{ }` tanpa diakhiri titik koma:

```
if (nilai == 80) {
    System.out.println("Nilainya 80");
}
```

4) **Identifier dalam Bahasa Java**
   Identifier adalah nama yang digunakan untuk menandai elemen dalam program, seperti variabel, kelas, metode, atau konstanta.
   Contoh:

```
double hasil = 33.33;
int totalPenjualan = 80;
```

Agar dapat dikenali dengan benar, identifier harus mengikuti aturan berikut:

1) Tidak boleh menggunakan keyword Java seperti `int`, `class`, atau `public`.
2) Dapat berisi huruf, angka, dan simbol underscore (_).
3) Tidak boleh diawali dengan angka.
4) Tidak boleh mengandung spasi.

Selain aturan dasar tersebut, terdapat pula konvensi penulisan yang umum digunakan oleh programmer Java:

1) Nama Class (PascalCase)
   Setiap kata diawali huruf besar.
   Contoh:
   `HelloBelajar, MahasiswaAktif, DataProdiSistemInformasi`

2) Nama Variabel dan Metode (camelCase)
   Kata pertama diawali huruf kecil, kata berikutnya huruf besar.
   Contoh:
   `totalNilai, hitungLuas(), getNamaMahasiswa()`

3) Nama Konstanta (SNAKE_CASE)
   Ditulis dengan huruf kapital dan dipisahkan oleh underscore.
   Contoh:
   `MAKSIMAL_NILAI, KURS_DOLLAR`

Memahami aturan dasar penulisan kode di Java sangat penting sebelum mempelajari konsep yang lebih kompleks. Kesalahan kecil seperti huruf kapital atau tanda baca dapat menyebabkan program gagal dijalankan. Dengan mengikuti kaidah ini, penulisan kode akan lebih rapi, mudah dibaca, dan sesuai standar yang berlaku di dunia pemrograman profesional.

4. Aturan Dasar Penulisan Kode Program PHP

Dalam mempelajari PHP sebagai bahasa pemrograman untuk pengembangan aplikasi web, memahami aturan dasar penulisan kode menjadi langkah awal yang sangat penting. Aturan ini menentukan bagaimana interpreter PHP membaca instruksi, mengenali struktur

program, serta mengeksekusi perintah. Dengan memahami kaidah tersebut sejak awal, kesalahan sintaks dapat diminimalkan dan kualitas kode menjadi lebih terjaga.

1) **Tag Pembuka dan Penutup PHP**
   Semua kode PHP harus dituliskan di dalam pasangan tag:
   ```
   <?php
       // kode PHP
   ?>
   ```
   Tag ini memberi tahu server bahwa bagian di dalamnya adalah instruksi PHP.
   Pada beberapa kasus, terutama pada file yang seluruhnya berisi PHP, tag penutup `(?>)` boleh dihilangkan untuk menghindari keluarnya whitespace yang tidak sengaja.

2) **Penggunaan Statement dan Titik Koma**
   PHP memiliki karakteristik unik mengenai huruf besar dan kecil.
   PHP tidak sensitif terhadap huruf besar pada *keyword* seperti `echo`, `if`, atau `while`. Namun, PHP sensitif terhadap huruf pada nama variabel dan sebagian identifier lainnya.
   Contoh:
   ```
   $nama = "Andi";
   echo $Nama; // ERROR, karena $nama ≠ $Nama
   ```
   Ini berarti `$data`, `$Data`, dan `$DATA` dianggap sebagai variabel yang berbeda.

3) **Struktur Blok Kode dengan Kurung Kurawal**
   Semua variabel dalam PHP harus diawali tanda dolar ($).
   Contoh:
   ```
   $umur = 20;
   $namaLengkap = "Laras";
   ```

Aturan dasar penamaan variabel dalam PHP:

- Tidak boleh diawali angka
  ✗ `$1nama`
  ✓ `$nama1`
- Tidak boleh mengandung spasi
  ✗ `$nama lengkap`
  ✓ `$nama_lengkap`
- Boleh menggunakan huruf, angka, dan underscore.

4) **Penggunaan Statement dan Titik Koma**

Setiap pernyataan (statement) dalam PHP harus diakhiri tanda titik koma (;).

Contoh:

```
$angka = 10;
echo $angka;
```

Tanpa titik koma, PHP akan menampilkan pesan kesalahan sintaks.

5) **Struktur Blok Kode dengan Kurung Kurawal**

Blok kode pada PHP ditandai dengan pasangan `{ }`, terutama untuk struktur kontrol seperti `if`, `for`, dan `while`.

Contoh:

```
if ($nilai > 70) {
    echo "Lulus";
}
```

Penggunaan kurung kurawal juga memengaruhi logika program, sehingga konsistensi sangat diperlukan.

6) **Penulisan Komentar dalam PHP**

Komentar berfungsi memberikan catatan atau penjelasan dalam kode. PHP mendukung tiga jenis komentar:

1) Single Line Comment

   Menggunakan // atau #:

   ```
   // Ini komentar satu defis
   ```

BAB 3
VARIABEL DAN TIPE DATA

A. Pendahuluan

Pemrograman modern memungkinkan komputer untuk mengolah data dalam berbagai bentuk, seperti angka, teks, dan nilai logika. Agar data tersebut dapat disimpan dan diproses, diperlukan variabel dan tipe data. Bab ini membahas konsep dasar yang menjadi pondasi utama dalam setiap bahasa pemrograman: Python, C++, Java, dan PHP.

B. Konsep Variabel

Variabel adalah nama yang diberikan pada sebuah lokasi memori di dalam komputer. Saat kita membuat variabel, sebenarnya kita sedang meminta komputer untuk menyediakan ruang penyimpanan dengan ukuran tertentu sesuai tipe data yang digunakan.
Setiap variabel memiliki tiga elemen:

1) Nama variabel (Identifier)
   Digunakan sebagai label untuk mengakses data. Nama variabel harus jelas, deskriptif, dan mengikuti aturan penamaan.
2) Tipe data
   Menentukan bentuk dan ukuran data yang disimpan.
3) Nilai
   Isi atau data aktual yang berada dalam variabel.

Contoh konsep variabel:

- `x` menyimpan angka
- `nama` menyimpan teks

Tabel 3. 1 Aturan Penamaan Variabel

Aturan	Contoh Salah	Contoh Benar
Tidak boleh diawali angka	`1nilai`	`nilai1`
Tidak boleh ada spasi	`nilai saya`	`nilai_saya`
Nama harus deskriptif	`x, y, z`	`jumlah_buku, totalHarga`

Tidak Boleh Menggunakan Simbol Khusus Tanda seperti @, %, &, *, -, dan lainnya. Kecuali underscore (boleh)	`nama-siswa`
	`data@1`
	`_`
		`nama_siswa`
		`data1`

Tidak Boleh Sama dengan Kata Kunci Bahasa	`for = 10`
	`int = 20`
	`return = "abc"`
		`Panjang = 5`
		`Ledef = 3`

Aturan penamaan variabel bertujuan agar kode:

- mudah dipahami,
- mudah diperbaiki,
- terlihat profesional,
- dan bebas error sintaks.

Python menggunakan dynamic typing (tidak perlu menentukan tipe data), sedangkan C++, Java, dan PHP memerlukan tipe data saat deklarasi (static/explicit typing).

Tabel 3. 2 Deklarasi Variabel pada Berbagai Bahasa

Bahasa	Contoh Deklarasi
Python	`nilai = 10`
C++	`int nilai = 10;`
Java	`int nilai = 10;`
PHP	`$nilai = 10;`

3. **Java**
   Java juga statis dan ketat dalam penentuan tipe data.

```
public class Main {
    public static void main(String[] args) {
      String nama = "Andi";
      int umur = 20;
      double tinggi = 165.5;
      boolean aktif = true;

      System.out.println("Nama: " + nama);
      System.out.println("Umur: " + umur);
      System.out.println("Tinggi: " + tinggi);
      System.out.println("Status aktif: " + aktif);
    }
}
```

4. **PHP**
   PHP mirip Python, bersifat dinamis, tipe variabel otomatis mengikuti nilai.

```
<?php
$nama = "Andi";
$umur = 20;
$tinggi = 165.5;
$aktif = true;

echo "Nama: $nama<br>";
echo "Umur: $umur<br>";
echo "Tinggi: $tinggi<br>";
echo "Status aktif: $aktif<br>";
?>
```

Tabel 3. 4 Contoh Tipe Data pada 4 Bahasa Pemrograman

Bahasa	Integer	Float / Double	String	Boolean
Python	`x = 10`	`y = 3.14`	`"Andi"`	`True`
C++	`int x = 10;`	`float y = 3.14;`	`"Andi"`	`true`
PHP	`$x = 10;`	`$y = 3.14;`	`"Andi"`	`true`
Java	`int x = 10;`	`double y = 3.14;`	`"Andi"`	`true`

Keterangan :

- Integer : Digunakan untuk menyimpan bilangan bulat tanpa koma, seperti 5, 50, atau -12. Tipe ini umum dipakai dalam operasi hitung, penomoran, atau perhitungan yang tidak memerlukan nilai desimal.
- Float / Double : Menyimpan bilangan pecahan atau desimal, misalnya 3.14 atau -0.75. Pada beberapa bahasa, seperti C++ dan Java, terdapat perbedaan antara *float* (presisi lebih kecil) dan *double* (presisi lebih tinggi).
- String : Merupakan kumpulan karakter yang membentuk teks. Contohnya "Andi", "Belajar Coding", atau "123" (meskipun terlihat seperti angka, tetap dianggap teks jika berada dalam tanda kutip).
- Boolean : Menampung nilai logika **true** atau **false**. Tipe ini sangat penting dalam pengambilan keputusan, seperti percabangan (if-else) atau perulangan.

2. Jenis-jenis Tipe Data dalam Bahasa Phyton

1) Tipe Data String dalam Python

Dalam Python, string digunakan untuk menyimpan data berbentuk teks seperti nama, alamat, kata, kalimat, hingga paragraf panjang, dan ukurannya tidak dibatasi secara khusus oleh bahasa Python, melainkan hanya oleh kapasitas memori sistem; pada Python 32-bit string dapat mencapai beberapa gigabyte, sedangkan pada Python 64-bit dapat mencapai puluhan gigabyte, sehingga kebutuhan pemrograman sehari-hari umumnya dapat terpenuhi tanpa masalah kapasitas.

```
a = 100
b = 30.23
c = 4j

print(a)
print(b)
print(c)
```

Jika kita ingin memastikan tipe data yang sedang digunakan, Python menyediakan fungsi bawaan `type()`:

```
a = 100
b = 30.23
c = 4j

print(type(a))
print(type(b))
print(type(c))
```

Dari sini terlihat bahwa Python secara otomatis menentukan tipe data sesuai nilai yang diberikan.

Batasan Angka dalam Python

Salah satu perbedaan mencolok Python dibandingkan bahasa lain adalah **integer tidak memiliki batas maksimum yang tetap**. Selama memori komputer masih tersedia, Python dapat menyimpan angka bulat yang sangat besar tanpa memunculkan kesalahan.

Contoh:

```
x = 999
y = -99999
z = 99999999999999999999

print(x)
print(y)
print(z)
```

Penulisan Angka dalam Bentuk Scientific Notation

Python juga mendukung penulisan angka desimal dalam format ilmiah atau *scientific notation*. Format ini sangat membantu untuk menuliskan angka yang terlalu besar atau terlalu kecil.

Dalam notasi matematika, misalnya:

125 ditulis sebagai 1.25×10^2

Di Python, pangkat 10 direpresentasikan dengan huruf **e** atau **E**.

Contoh:

```
x = 3e2       # 3 × 10²
y = 3e-2      # 3 × 10⁻²
z = 1.34E5    # 1.34 × 10⁵

print(x)
print(y)
print(z)
```

Keluaran:

```
300.0
0.03
134000.0
```

Perhatikan bahwa meskipun 3e2 seolah-olah merupakan angka bulat (300), Python tetap menganggapnya sebagai **float**, sehingga keluaran yang muncul akan berakhiran .0.

3) **Tipe Data Boolean dalam Bahasa Python**

Tipe data Boolean adalah tipe data yang hanya memiliki dua kemungkinan nilai, yaitu: **True dan False.** Boolean digunakan untuk menyatakan kondisi benar atau salah. Tipe data ini sangat penting dalam pembuatan logika program, terutama pada percabangan seperti **if**, **elif**, dan **else**.

Penulisan Boolean dalam Python

Penulisan nilai Boolean harus menggunakan huruf pertama **kapital**:

```
abc = True
def = False
print(abc)
print(def)
```

Jika ditulis dengan huruf kecil seperti true atau false, Python akan menghasilkan error karena dianggap sebagai nama variabel yang tidak dikenal.

Boolean dari Operasi Perbandingan

Selain ditulis langsung, Boolean sering muncul sebagai hasil operasi perbandingan. Contoh umum:

```
print(12 < 10)    # False
print(12 > 10)    # True
print("A" == "a") # False
```

Setiap operasi perbandingan akan menghasilkan **`True`** atau **False**, tergantung benar atau tidaknya kondisi tersebut.

Penggunaan Boolean dalam Logika Program

Boolean biasanya dipakai untuk menentukan alur program, seperti pada struktur **`if`**:

```
a = 12
b = 10

if a < b:
    print("a lebih kecil dari b")
elif a > b:
    print("a lebih besar dari b")
else:
    print("a sama dengan b")
```

Pada contoh tersebut, Python mengevaluasi ekspresi `a > b` sebagai **`True`**, sehingga perintah pada blok tersebut dijalankan.

4) **Tipe Data List dalam Bahasa Python**

List adalah salah satu tipe data Python yang digunakan untuk menyimpan kumpulan nilai. List mirip dengan array pada bahasa pemrograman lain, tetapi jauh lebih fleksibel karena:

- Bisa berisi berbagai jenis data (string, angka, boolean, dll.)

- Ukurannya dapat berubah (dinamis)
- Mendukung banyak operasi untuk memodifikasi dan mengolah data

List termasuk salah satu dari empat struktur data utama Python: **List, Tuple, Set, dan Dictionary**.

Cara Membuat List

List ditulis menggunakan **tanda kurung siku** [ ] dan setiap elemen dipisahkan dengan koma.

Contoh:

```
abc= ["Belajar", "Python", "Pemrograman"]
def = [100, 200, 300, 400]
ghi = ["Python", 200, 6.99, True]

print(abc)
print(def)
print(ghi)
```

Mengakses Elemen List

Setiap elemen dalam List memiliki **index**. Python menggunakan index mulai dari angka **0**.

```
abc= ["Python", 200, 6.99, True]

print(abc[0])   # Python
print(abc[1])   # 200
print(abc[2])   # 6.99
print(abc[3])   # True
```

Mengubah Nilai dalam List

Elemen List dapat diganti dengan nilai defu menggunakan index.

```
abc= ["Python", 200, True]

abc[0] = "Belajar"
abc[2] = False
print(abc)
```

Mengakses Elemen Tuple

Cara mengakses elemen Tuple sama seperti List, yaitu menggunakan nomor indeks dimulai dari 0.

```
abc= ("Python", 200, 6.99, True, "Belajar")
print(abc[0])
print(abc[1])
print(abc[2])
```

Mengambil Beberapa Elemen (Slicing)

Tuple juga mendukung *slicing*, yaitu mengambil sebagian isi berdasarkan rentang indeks.

```
abc= ("Python", 200, 6.99, True, "Belajar")
print(abc[1:4])   # index 1 sampai 3
print(abc[:3])    # awal sampai index 2
print(abc[2:])    # mulai index 2 sampai akhir
print(abc[:])     # tampilkan semua
```

Mengapa Tuple Tidak Bisa Diubah?

Tuple bersifat *immutable*. Jika mencoba mengganti salah satu elemennya, Python akan menolak.

```
abc= ("Python", 200, 6.99)
abc[0] = "Belajar"
```

Hasil:

```
TypeError: 'tuple' object does not support item assignment
```

Artinya elemen dalam Tuple tidak dapat diganti atau ditambah.

Kapan Menggunakan Tuple?

Gunakan **Tuple** jika datanya bersifat tetap, misalnya:

- koordinat (x, y)
- data konfigurasi yang tidak berubah
- daftar hari dalam seminggu
- data yang tidak ingin dimodifikasi secara tidak sengaja

Gunakan **List** jika datanya perlu diperdefui sepanjang program.

6) **Tipe Data Set dalam Bahasa Python**

Tipe data **set** dalam Python adalah struktur data yang digunakan untuk menyimpan kumpulan nilai unik dan sering dipakai dalam operasi yang berhubungan dengan himpunan. Berbeda dengan tipe data lain seperti **list** atau **tuple**, tipe data set memiliki beberapa karakteristik khusus:

- Tidak memiliki indeks
- Tidak memiliki urutan tetap
- Setiap nilai harus unik (unique value)

Karena tidak memiliki indeks, posisi elemen dalam set tidak dapat diprediksi. Nilai yang dimasukkan bisa muncul di urutan berbeda ketika ditampilkan. Selain itu, Python akan secara otomatis menghapus nilai yang duplikat.

Cara Membuat Tipe Data Set

Set dibuat menggunakan tanda **kurung kurawal {}** dan setiap elemen dipisahkan oleh koma.

Contoh:

```
abc= {"Belajar", "Python", "di", "Duniailkom"}
def = {100, 200, 300, 400}
ghi = {"Python", 200, 6.99, True}

print(abc)
print(def)
print(ghi)
```

Perhatikan bahwa urutannya berubah karena set **tidak terurut**.

Perbedaan Set dengan List dan Tuple

Untuk menunjukkan perbedaan penulisan ketiga tipe data tersebut, gunakan fungsi type().

```
abc= ["Belajar", "Python", "di", "Duniailkom"]
print(type(abc))   # list

abc= ("Belajar", "Python", "di", "Duniailkom")
print(type(abc))   # tuple
```

```
abc= {"Belajar", "Python", "di", "Duniailkom"}
print(type(abc))   # set
```

Sifat Tipe Data Set

1) Tidak mendukung indeks
   Karena tidak memiliki indeks, elemen set tidak dapat diakses menggunakan [index].
   Contoh:
   ```
   abc= {"Belajar", "Python", "di", "Kampus"}

   print(abc[1])
   ```
   Hasil error:
   ```
   TypeError: 'set' object does not support indexing
   ```

2) Tidak menyimpan nilai ganda
   Jika sebuah nilai dimasukkan lebih dari satu kali, set hanya menyimpannya sekali.
   ```
   abc=     {"Belajar",     "Python",     "di",
   "Duniailkom", "di"}
   def = {100, 200, 300, 400, 200, 300}

   print(abc)
   print(def)
   ```
   Hasil:
   ```
   {'Python', 'di', 'Duniailkom', 'Belajar'}
   {200, 100, 400, 300}
   ```
   Duplikasi otomatis dihilangkan.

Operasi Himpunan pada Set

Set sangat cocok digunakan untuk operasi himpunan seperti gabungan (union) dan irisan (intersection)**.**

Contoh:

```
abc= {1, 2, 3, 4, 5}
def = {3, 4, 5, 6, 7}

print(abc| def)  # union
print(abc& def)  # intersection
```

Hasil:

```
{1, 2, 3, 4, 5, 6, 7}
{3, 4, 5}
```

Penjelasan:

- Union (|) → menggabungkan kedua himpunan tanpa duplikasi.
- Intersection (**&**) → menampilkan nilai yang muncul di kedua set.

7) **Tipe Data Dictionary dalam Bahasa Python**

Tipe data **dictionary** adalah struktur data di Python yang terdiri dari pasangan **key–value**. Berbeda dengan array biasa yang indeksnya berupa angka, dictionary memungkinkan key berbentuk **string**, **angka**, atau tipe data lain yang bersifat immutable. Dalam beberapa bahasa pemrograman lain (misalnya PHP), dictionary sering disebut **associative array**.

Format dasar dictionary:

```
nama_variabel = {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
}
```

Dictionary sangat cocok digunakan untuk menyimpan data yang kompleks dan terstruktur.

3. Jenis-jenis Tipe Data dalam Bahasa C++

1) Tipe Data Char Bahasa C++

Tipe data **char** digunakan untuk menyimpan **sebuah karakter tunggal**, baik berupa huruf, angka, maupun simbol. Setiap variabel bertipe `char` menggunakan **1 byte memori**. Nilai karakter yang disimpan direpresentasikan menggunakan kode **ASCII**.

Contoh karakter yang valid pada tipe `char`:`'A'`, `'z'`, `'5'`, `'?'`, `'#'`, `'*'`

Contoh Penggunaan Tipe Data `char`

Berikut contoh program sederhana yang menyimpan sebuah karakter ke dalam variabel bertipe `char`:

```
#include <iostream>
using namespace std;

int main() {
    char huruf;
    huruf = 'D';

    cout << "Huruf yang tersimpan adalah: " <<
huruf << endl;

    return 0;
}
```

Program di atas mendefinisikan variabel `huruf` bertipe `char`, kemudian diisi dengan nilai `'D'` dan ditampilkan menggunakan `cout`.

Tipe Data `char` sebagai Konstanta

Tipe `char` juga dapat digunakan sebagai **konstanta** dengan kata kunci `const`.

```
#include <iostream>
using namespace std;

int main() {
    const char huruf = 'a';
```

```
    cout << "Huruf yang tersimpan adalah: " <<
huruf << endl;

    return 0;
}
```

Karena bertipe konstanta, nilai `huruf` tidak dapat diubah setelah didefinisikan.

Menggunakan `cin` untuk Input Karakter

Contoh berikut memperlihatkan bagaimana membaca banyak karakter melalui input:

```
#include <iostream>
using namespace std;

int main() {
    char karakter1, karakter2, karakter3;

    cout << "Input 3 karakter sembarang" <<
endl;
    cout << "==========================" << endl
<< endl;

    cout << "Karakter pertama: ";
    cin >> karakter1;

    cout << "Karakter kedua: ";
    cin >> karakter2;

    cout << "Karakter ketiga: ";
    cin >> karakter3;

    cout << endl;

    cout << "Karakter yang diinput adalah: "
         << karakter1 << ", "
```

```
            << karakter2 << ", dan "
            << karakter3 << endl;

    return 0;
}
```

Program ini meminta tiga karakter dari pengguna, kemudian menampilkannya kembali.

Catatan Penting tentang Tipe char

- Tipe char hanya dapat menyimpan **satu karakter**.
- Karakter dapat berupa huruf, angka, atau simbol khusus.
- Jika Anda memasukkan angka, angka tersebut akan dianggap sebagai karakter, bukan sebagai nilai numerik (contoh: '5' berbeda dengan angka 5).
- Untuk data yang berisi lebih dari satu karakter seperti kata atau kalimat, gunakan tipe data string, bukan char.

Pembahasan mengenai tipe data string akan dijelaskan pada bagian selanjutnya.

2) **Tipe Data Integer Bahasa C++**

Dalam C++, tipe data *integer* digunakan untuk menyimpan bilangan bulat baik positif maupun negatif, misalnya **12**, **0**, atau **-450**. Meski sama-sama mewakili angka bulat, C++ menyediakan beberapa varian integer dengan ukuran memori dan rentang nilai yang berbeda.

Macam-Macam Tipe Integer

Secara umum terdapat empat jenis integer utama: `char`, `short`, `int`, `long`. Meskipun `char` sering dianggap sebagai tipe untuk karakter, sebenarnya di level memori ia disimpan sebagai angka. Itulah sebabnya char juga termasuk kelompok integer. Perbedaan karakter dan angka baru terlihat saat nilai tersebut ditampilkan.

Ukuran Memori dan Rentang N

Setiap tipe integer memiliki kapasitas penyimpanan yang berbeda. Semakin besar ukuran memorinya, semakin luas pula rentang nilai yang bisa disimpan.

- Tipe Data Signed (bertanda)
  Mampu menyimpan nilai positif dan negatif.

Tabel 3. 5 Tipe Data Signed pada C++

Tipe Data	Ukuran Memori	Rentang Nilai
`char`	1 byte	-128 s.d. 127
`short`	2 bytes	-32.768 s.d. 32.767
`int` (2 byte)	2 bytes	-32.768 s.d. 32.767
`int` (4 byte)	4 bytes	-2.147.483.648 s.d. 2.147.483.647
`long`	4 bytes	sama seperti int 4 byte

Catatan: ukuran int bisa berbeda tergantung compiler dan arsitektur komputer.

- Tipe Data Unsigned
  Jika suatu tipe integer diberi kata kunci **unsigned**, maka tipe tersebut tidak menyimpan nilai negatif, sehingga kapasitas untuk angka positif menjadi dua kali lebih besar.

Tabel 3. 6 Tipe Data Unsigned pada C++

Tipe Data	Ukuran Memori	Rentang Nilai
`unsigned char`	1 byte	0 s.d. 255
`unsigned short`	2 bytes	0 s.d. 65.535
`unsigned int` (2 byte)	2 bytes	0 s.d. 65.535
`unsigned int` (4 byte)	4 bytes	0 s.d. 4.294.967.295
`unsigned long`	4 bytes	sama seperti unsigned int 4 byte

Tipe unsigned cocok untuk data yang tidak mungkin negatif, seperti jumlah barang, umur, tinggi badan, dan sebagainya.

Contoh Program: Penggunaan Integer

Contoh 1 — Penggunaan short, int, dan long

```
#include <iostream>
using namespace std;

int main() {
    short angka1;
    int   angka2;
    long  angka3;

    angka1 = 32767;
    angka2 = 2147483647;
    angka3 = 2147483647;

    cout << "Isi angka1 (short): " << angka1 <<
endl;
    cout << "Isi angka2 (int)  : " << angka2 <<
endl;
    cout << "Isi angka3 (long) : " << angka3 <<
endl;

    return 0;
}
```

Contoh 2 — Perlakuan Khusus pada char

Tipe `char` dapat berisi angka maupun karakter karena nilai karakter disimpan dalam bentuk *kode ASCII*.

```
#include <iostream>
using namespace std;

int main() {
    char foo = 65;
    char bar = 'Z';

    cout << "Isi foo: " << foo << endl;
```

```
int main()
{
    double var1 = 3.1415926;

    cout << fixed;
    cout << setprecision(2);

    cout << var1 << endl;   // 3.14
}
```

Output:

```
3.14
```

Pengaturan setprecision() berlaku terus sampai diubah kembali.

Variasi Setiap Baris

```
cout << setprecision(0) << var1 << endl;
cout << setprecision(1) << var1 << endl;
cout << setprecision(2) << var1 << endl;
cout << setprecision(3) << var1 << endl;
cout << setprecision(4) << var1 << endl;
```

Output:

```
3
3.1
3.14
3.142
3.1416
```

Menampilkan Float/Double dengan printf

C juga menyediakan printf() yang lebih fleksibel:

```
printf("Isi var1: %f\n", var1);
printf("Isi var1: %15f\n", var1);
printf("Isi var1: %015f\n", var1);
printf("Isi var1: %.3f\n", var1);
printf("Isi var1: %010.3f\n", var1);
```

Pada saat ditampilkan menggunakan `cout`, nilai true ditampilkan sebagai 1, sedangkan false ditampilkan sebagai 0. Hal ini karena dalam level mesin, boolean dipetakan ke angka biner: 1 untuk benar dan 0 untuk salah.
Catatan penting:
Nilai true dan false harus ditulis dalam huruf kecil. Penulisan seperti True, FALSE, atau variasi lain akan dianggap tidak valid.

Contoh Kesalahan Penulisan Nilai Boolean

```
#include <iostream>
using namespace std;

int main() {
    bool var1 = True;
    bool var2 = false;

    cout << var1 << endl;
    cout << var2 << endl;

    return 0;
}
```

Output Error

```
error: 'True' was not declared in this scope
```

Kesalahan ini terjadi karena C++ tidak mengenali True sebagai kata kunci boolean yang sah.

Boolean dari Operasi Perbandingan

Dalam praktiknya, nilai boolean lebih sering diperoleh dari operasi perbandingan, bukan ditulis langsung oleh programmer.
Operasi seperti:

- < (lebih kecil)
- (lebih besar)
- == (sama dengan)
- != (tidak sama dengan)
- <=, >=

akan menghasilkan nilai true atau false.

Contoh Program

```
#include <iostream>
using namespace std;

int main() {
    bool var1, var2, var3;

    var1 = 12 < 10;
    var2 = 30 > 25;
    var3 = 'A' == 'a';

    cout << "Isi var1 = " << var1 << endl;
    cout << "Isi var2 = " << var2 << endl;
    cout << "Isi var3 = " << var3 << endl;

    return 0;
}
```

Hasil Output

```
Isi var1 = 0
Isi var2 = 1
Isi var3 = 0
```

Penjelasan singkat:

- 12 < 10 → salah → false → ditampilkan sebagai 0
- 30 > 25 → benar → true → ditampilkan sebagai 1
- 'A' == 'a' → huruf berbeda → false

Boolean dalam Struktur Kendali (if-else)

Boolean menjadi inti logika ketika menggunakan struktur percabangan.

Contoh

```
#include <iostream>
using namespace std;

int main() {
    int var1 = 12;
    int var2 = 10;
```

```
    if (var1 < var2) {
        cout << "var1 lebih kecil daripada var2"
<< endl;
    }
    else if (var1 > var2) {
        cout << "var1 lebih besar daripada var2"
<< endl;
    }
    else {
        cout << "var1 sama dengan var2" << endl;
    }

    return 0;
}
```

Hasil Output

```
var1 lebih besar daripada var2
```

Percabangan di atas bekerja berdasarkan hasil perbandingan:

- Ekspresi var1 < var2 → false
- Ekspresi var1 > var2 → true → blok tersebut dijalankan

Inilah alasan mengapa tipe boolean sangat berperan dalam logika program.

5) **Tipe Data String Bahasa C++**

Dalam pemrograman, string adalah tipe data yang digunakan untuk menyimpan kumpulan karakter. Contohnya berupa kata seperti "aku", "kamu", hingga kalimat lengkap seperti "Sedang belajar bahasa C++". Di dalam C++, tipe data string dapat dibuat menggunakan keyword string, dan setiap teks harus dituliskan di antara tanda kutip dua (" ")

Menggunakan Karakter Khusus (Escape Characters)
String dapat berisi karakter khusus seperti:

- \" → menampilkan tanda kutip dua
- \' → menampilkan tanda kutip satu
- \n → membuat baris baru

Contoh

```
#include <iostream>
using namespace std;

int main() {
    string teks = "Belajar \nC++ \ndi \"kelas pemrograman\"";

    cout << teks << endl;

    return 0;
}
```

Output

```
Belajar
C++
di "kelas pemrograman"
```

Karakter \n menyebabkan teks berikutnya berpindah ke baris baru.

String sebagai Deretan Karakter
Dalam C++, string dapat diperlakukan seperti array yang berisi karakter. Setiap karakter dapat diakses dengan indeks mulai dari 0.

Contoh

```
#include <iostream>
using namespace std;

int main() {
    string kata = "Belajar C++";

    cout << kata[0] << endl;
    cout << kata[1] << endl;
```

Catatan Tentang Method pada String
Metode seperti `length()` dan `size()` adalah fungsi yang melekat pada objek string.
Cara pemanggilannya menggunakan tanda titik:
`variabelString.method()`

Contoh:
`kalimat.length()` mengembalikan jumlah karakter dari variabel kalimat.

Topik terkait method, function, dan object akan dibahas lebih rinci di bab selanjutnya.

6) **Tipe Data Array Bahasa C++**
Array adalah struktur data berisi sekumpulan nilai dengan tipe yang sama, misalnya kumpulan bilangan bulat, karakter, atau data pecahan. Dalam konteks bahasa Indonesia, array sering juga disebut larik. Dengan array, penyimpanan data menjadi lebih teratur dan efisien, terutama ketika data yang dikelola jumlahnya banyak. Setiap elemen array memiliki posisi tertentu yang disebut indeks.

Contoh tanpa array (5 nilai siswa):

```
int nilai1, nilai2, nilai3, nilai4, nilai5;
```

Contoh dengan array:

```
int nilai[5];
```

Jika data yang disimpan lebih besar, seperti 1000 nilai, cukup:

```
int nilai[1000];
```

Bentuk Penulisan Array di C++
Format dasar deklarasi array adalah:
`tipe_data nama_array[jumlah_elemen];`

Contoh deklarasi array berisi 100 bilangan pecahan:
`float bilangan[100];`

Mengakses Nilai di Dalam Array
Untuk membaca atau mengisi elemen array, digunakan indeks.
`bilangan[5] = 3.14;`

Namun penting diingat bahwa indeks pertama array dimulai dari 0, bukan dari 1.

- Elemen pertama → `bilangan[0]`
- Elemen ke-100 → `bilangan[99]`

Skema ini juga digunakan di hampir semua bahasa modern seperti Java, PHP, atau JavaScript.

Contoh Program Array (Integer)

```
#include <iostream>
using namespace std;

int main() {
    int nilai[5];

    nilai[0] = 23;
    nilai[1] = 50;
    nilai[2] = 34;
    nilai[3] = 78;
    nilai[4] = 90;

    cout << "Nilai ke-1: " << nilai[0] << endl;
    cout << "Nilai ke-2: " << nilai[1] << endl;
    cout << "Nilai ke-3: " << nilai[2] << endl;
    cout << "Nilai ke-4: " << nilai[3] << endl;
    cout << "Nilai ke-5: " << nilai[4] << endl;

    return 0;
}
```

Mengisi Array Secara Langsung

Nilai array juga dapat diberikan saat deklarasi:

```
char kode[5] = {'D', '$', 'o', 'Z', '8'};
```

Kemudian diakses seperti biasanya:

```
kode[0], kode[1], ..., kode[4]
```

Deklarasi Array Tanpa Menulis Jumlah Elemen

C++ dapat otomatis menentukan jumlah elemen dari banyaknya nilai awal:

```
char simbol[] = {'D', '$', 'o', 'Z', '8'};
```

Syarat metode ini:

✓ jumlah elemen harus langsung dituliskan

✗ tidak boleh digunakan jika pengisian nilai dilakukan belakangan

Contoh yang salah:

```
int nilai[]; // error
nilai[0] = 23;
```

Mengubah Elemen Array

Setiap elemen array bersifat dapat diubah selama program berjalan:

```
float angka[] = {3.14, -99.01, 0.002};

angka[1] = 9.123;
angka[2] = 12.9925;
```

Hubungan Array dan String

Dalam C++, string dapat diakses seperti array karakter:

```
string teks = "Belajar PHP";
teks[8] = 'C';
teks[9] = '+';
teks[10] = '+';

cout << teks;  // hasil: Belajar C++
```

Setiap indeks pada string dapat dimodifikasi selama tidak melewati batas panjang string.

Struct dengan Input dari Pengguna

Struct juga dapat diisi melalui input langsung dari pengguna.

```
#include <iostream>
using namespace std;

struct Siswa {
    string nama;
    string sekolah;
    unsigned int uangSaku;
};

int main() {
    Siswa siswa01;

    cout << "== Pendaftaran Siswa ==" << endl;

    cout << "Nama: ";
    getline(cin, siswa01.nama);

    cout << "Sekolah: ";
    getline(cin, siswa01.sekolah);

    cout << "Uang saku: ";
    cin >> siswa01.uangSaku;

    cout << endl;

    cout << siswa01.nama << " bersekolah di " <<
siswa01.sekolah
         << " dengan uang saku " <<
siswa01.uangSaku << endl;

    return 0;
}
```

Program ini meminta pengguna memasukkan data, kemudian menampilkannya kembali menggunakan anggota struct.

8) **Tipe Data Enum Bahasa C++**

Tipe data enum (enumerated) adalah tipe data khusus yang berisi daftar nilai yang kita tentukan sendiri. Contohnya, kita dapat membuat enum warna yang hanya memiliki pilihan merah, kuning, biru, dan ungu, atau enum nama_hari yang berisi daftar nama hari. Setelah tipe enum didefinisikan, variabel yang menggunakan tipe tersebut hanya dapat diisi dengan nilai yang sudah terdaftar, tidak bisa dengan nilai lain.

Format Penulisan Enum

Enum dideklarasikan menggunakan keyword enum. Format dasarnya:

```
enum nama_tipe {nilai1, nilai2, nilai3, ... };
```

Contoh enum untuk nama hari:

```
enum nama_hari {minggu, senin, selasa, rabu,
kamis, jumat, sabtu};
```

Agar dapat digunakan, enum harus dihubungkan dengan sebuah variabel di dalam fungsi `main()`.

Contoh Program Penggunaan Enum

1. Penggunaan Dasar

```
#include <iostream>
using namespace std;

enum nama_hari {minggu, senin, selasa, rabu,
kamis, jumat, sabtu};

int main()
{
    nama_hari var1;

    var1 = senin;
    cout << var1 << endl;
```

```
    var1 = rabu;
    cout << var1 << endl;

    var1 = sabtu;
    cout << var1 << endl;

    return 0;
}
```

Output:
1
3
6

Penjelasan: nilai yang ditampilkan adalah indeks dari setiap item enum (dimulai dari 0).

- minggu → 0
- senin → 1
- sabtu → 6

Kesalahan yang Bisa Terjadi

Enum tidak menerima nilai di luar daftar. Contoh berikut menyebabkan error:

```
var1 = monday;  // error: 'monday' was not
declared in this scope
```

Karena *monday* tidak ada dalam daftar enum, maka program tidak dapat mengaksesnya.

Mengganti Nilai Default Enum

Secara bawaan, setiap item enum memiliki nilai berupa nomor indeks. Namun, kita bisa mengganti nilainya sendiri:

Tidak seperti bahasa C/C++, Java tidak memiliki tipe integer unsigned, sehingga seluruh tipe integer di Java selalu menyediakan nilai negatif dan positif.

Contoh Program Penggunaan Integer di Java

```
class BelajarJava {
  public static void main(String args[]){

    byte  varByte;
    short varShort;
    int   varInt;
    long  varLong;

    varByte = 120;
    varShort = 32000;
    varInt = 1000000000;
    varLong = 10000000000000000L;//tambahkan huruf L

    System.out.println("varByte = " + varByte);
    System.out.println("varShort = " + varShort);
    System.out.println("varInt = " + varInt);
    System.out.println("varLong = " + varLong);
  }
}
```

Catatan penting

Angka yang disimpan dalam variabel bertipe long wajib diakhiri dengan huruf L.

Jika tidak, Java menganggapnya sebagai int, sehingga nilai yang terlalu besar akan menghasilkan error integer number too large.

Literal Bilangan Biner, Oktal, Desimal, dan Heksadesimal

Java memungkinkan penulisan angka dalam beberapa basis penomoran:

- Biner: diawali 0b → contoh: 0b10100100
- Oktal: diawali 0 → contoh: 0244
- Desimal: angka biasa → contoh: 164
- Heksadesimal: diawali 0x → contoh: 0xA4

```
class BelajarJava {
  public static void main(String args[]){

    int v1 = 0b10100100;   // biner
    int v2 = 0244;         // oktal
    int v3 = 164;          // desimal
    int v4 = 0xA4;         // heksadesimal

    System.out.println(v1);
    System.out.println(v2);
    System.out.println(v3);
    System.out.println(v4);
  }
}
```

Keempatnya akan menghasilkan nilai 164 ketika dicetak, karena Java akan mengonversinya ke bentuk desimal.

Membaca Input Integer dengan Scanner

Untuk menerima input dari pengguna, Java menggunakan class Scanner.Masing-masing tipe integer memiliki metode pembacaan tersendiri:

- `nextByte()`
- `nextShort()`
- `nextInt()`
- `nextLong()`

Contoh:

```
import java.util.Scanner;

class BelajarJava {
  public static void main(String args[]){

    Scanner input = new Scanner(System.in);

    byte  b;
    short s;
    int   i;
    long  l;

    System.out.print("Masukkan byte: ");
    b = input.nextByte();

    System.out.print("Masukkan short: ");
    s = input.nextShort();

    System.out.print("Masukkan int: ");
    i = input.nextInt();

    System.out.print("Masukkan long: ");
    l = input.nextLong();

    System.out.println();
    System.out.println("=== Hasil ===");
    System.out.println("byte  = " + b);
    System.out.println("short = " + s);
    System.out.println("int   = " + i);
    System.out.println("long  = " + l);
  }
}
```

2) **Tipe Data Float dan Double Bahasa Pemrograman Java**

Dalam Java, nilai pecahan (bilangan desimal) disimpan menggunakan dua tipe data utama, yaitu float dan double. Keduanya dipakai untuk merepresentasikan angka dengan bagian pecahan, misalnya: `3.14`, `12.0`, atau `-0.025`. Seperti bahasa pemrograman lain, Java menggunakan titik (.) sebagai pemisah decimal bukan koma seperti penulisan sehari-hari.

Perbedaan Float dan Double

Java menyediakan dua tipe berbeda agar programmer bisa memilih sesuai kebutuhan ketelitian dan ruang memori.

Tabel 3. 9 Perbedaan Float dan Double pada Java

Tipe Data	Ukuran Memori	Jangkauan Nilai
float	4 byte (32 bit)	$\pm 3.4 \times 10^{-38}$ sampai 3.4×10^{38}
double	8 byte (64 bit)	$\pm 1.7 \times 10^{-308}$ sampai 1.7×10^{308}

Walaupun rentangnya sangat besar, tipe data pecahan tidak selalu presisi 100%, karena penyimpanan bilangan real di komputer menggunakan representasi biner floating point.

Contoh Program Float dan Double

```
class BelajarJava {
  public static void main(String[] args){

    float  nilaiFloat;
    double nilaiDouble;

    nilaiFloat = 234.45F;   // harus pakai akhiran F
    nilaiDouble = 234.45;    // default-nya dianggap
double

    System.out.println("Float  = " + nilaiFloat);
    System.out.println("Double = " + nilaiDouble);
  }
}
```

Penjelasan :

- Variabel `nilaiFloat` harus diisi dengan angka yang diakhiri huruf `F (234.45F)`, karena literal desimal di Java otomatis dianggap double.
- Jika akhiran `F` tidak ditulis, compiler akan menampilkan pesan bahwa terjadi potensi kehilangan presisi *(lossy conversion).*

Kesalahan Umum (Tidak Menambah "F")

```
float var1;
var1 = 234.45;   // error
```

Error yang muncul:

```
incompatible types: possible lossy conversion from
double to float
```

Notasi Ilmiah (Scientific Notation)

Java juga mendukung penulisan angka pecahan dalam bentuk pangkat 10 menggunakan e atau E.

Contoh:

```
float  a = 3.12e2F;  // 3.12 × 10² = 312.0
double b = 4E-3;     // 4 × 10⁻³ = 0.004

System.out.println(a);  // 312.0
System.out.println(b);  // 0.004
```

Input Float dan Double dengan Scanner

Untuk membaca nilai pecahan dari keyboard, digunakan metode:

- `nextFloat()`
- `nextDouble()`

Penggunaan Locale disarankan agar pembacaan angka memakai titik.

```
import java.util.Scanner;
import java.util.Locale;
```

```
class BelajarJava {
  public static void main(String[] args){

    Scanner input = new Scanner(System.in).useLocale
(Locale.US);

    float  nilaiFloat;
    double nilaiDouble;

    System.out.print("Masukkan nilai float : ");
    nilaiFloat = input.nextFloat();

    System.out.print("Masukkan nilai double: ");
    nilaiDouble = input.nextDouble();

    System.out.println("\n--- Hasil ---");
    System.out.println("Float  = " + nilaiFloat);
    System.out.println("Double = " + nilaiDouble);
  }
}
```

Kenapa harus pakai Locale.US?

Beberapa komputer menggunakan koma sebagai pemisah desimal, yang bisa membuat pembacaan angka pecahan gagal. Dengan mengatur locale ke US, Java memastikan pemisah yang digunakan adalah titik.

Memilih `Float` atau `Double`?

- Untuk kebanyakan kasus, gunakan `double` karena: lebih presisi, jangkauannya lebih besar dan literal pecahan default Java adalah double, sehingga lebih praktis.
- Gunakan `float` hanya jika: sedang menghemat memori (misal: pemrosesan grafik), atau API/library tertentu memang meminta tipe float.

3) **Tipe Data Boolean Bahasa Pemrograman Java**

Tipe data boolean digunakan untuk menyimpan nilai logika yang hanya memiliki dua kemungkinan: true atau false. Nilai ini sangat penting dalam pemrograman karena menjadi dasar pengambilan keputusan pada struktur percabangan, seperti if, else, atau switch. Konsep boolean sering digunakan untuk memeriksa suatu kondisi. Misalnya, memastikan apakah sebuah angka genap, apakah suatu nilai lebih besar dari nilai lain, atau apakah dua karakter sama.

Contoh Penggunaan Boolean Sederhana

Untuk membuat variabel boolean, kita menggunakan keyword boolean:

```
class BelajarJava {
  public static void main(String[] args){

    boolean var1;
    boolean var2;

    var1 = true;
    var2 = false;

    System.out.println("var1 = " + var1);
    System.out.println("var2 = " + var2);
  }
}
```

Hasil:

```
var1 = true
var2 = false
```

Catatan penting : Nilai boolean harus ditulis dengan huruf kecil:

- true ✓
- false ✓
- `True, FALSE, False` ✗ (akan dianggap variabel yang tidak dikenal)

Karakter Khusus (Escape Character)

Karakter tertentu tidak bisa ditulis langsung dan harus memakai escape character, yaitu backslash `(\)`:

Escape	Arti
`\'`	Tanda kutip satu
`\"`	Tanda kutip dua
`\n`	Baris baru

Contoh:

```
class BelajarJava {
  public static void main(String args[]){

    char var1 = '\'';
    char var2 = '\n';
    char var3 = '\"';

    System.out.print(var1);
    System.out.print(var2);
    System.out.print(var3);
  }
}
```

Membaca Input char dengan Scanner

Scanner tidak memiliki metode khusus seperti nextChar().
Karena itu, pembacaan karakter dilakukan dari string pertama, lalu diambil karakter indeks 0.

```
import java.util.Scanner;

class BelajarJava {
  public static void main(String args[]){

    Scanner input = new Scanner(System.in);

    char var1, var2;

    System.out.print("var1: ");
```

```
    var1 = input.next().charAt(0);

    System.out.print("var2: ");
    var2 = input.next().charAt(0);

    System.out.println();
    System.out.println("## Hasil ##");
    System.out.println("var1 = " + var1);
    System.out.println("var2 = " + var2);
  }
}
```

Pengguna cukup memasukkan satu karakter, dan karakter pertama dari input tersebut akan diproses sebagai char.

5) **Tipe Data String Bahasa Pemrograman Java**

Dalam pemrograman, string merupakan tipe data yang digunakan untuk menyimpan sekumpulan karakter. Nilainya dapat berupa kata, kalimat, atau gabungan karakter seperti `"Halo", "Belajar Java",` atau kalimat panjang lainnya. Di Java, string dibuat dengan menuliskan teks di dalam tanda kutip dua `(" ")` dan menggunakan tipe String.

Contoh Deklarasi Variabel String

```
class BelajarJava {
  public static void main(String args[]){

    String var1 = "Belajar Java";
    String var2 = " di Kampus ini";

    System.out.print(var1);
    System.out.print(var2);
  }
}
```

Output:

```
Belajar Java di Kampus ini
```

Pada contoh tersebut, dua variabel string digabungkan menggunakan System.out.print() yang menampilkan teks secara berurutan.

Penggabungan String (Concatenation)
String dapat disambung menggunakan operator +:

```
class BelajarJava {
  public static void main(String args[]){

    String var1 = "Belajar Java";
    String var2 = " dari dasar";

    System.out.print(var1 + var2);
  }
}
```

Output:

```
Belajar Java dari dasar
```

Proses penggabungan ini disebut concatenation dan sangat sering digunakan untuk menyusun kalimat.

Menggunakan Karakter Khusus dalam String
Di dalam string, karakter tertentu membutuhkan escape sequence, misalnya:

- `\n` untuk baris baru,
- `\"` untuk tanda kutip dua,
- `\'` untuk tanda kutip satu.

Contoh:

```
class BelajarJava {
  public static void main(String args[]){

    String var1 = "Belajar \nJava \ndi \"kelas\"";
```

```
    System.out.print(var1);
  }
}
```

Output:

```
Belajar
Java
di "kelas"
```

Walaupun ditulis dalam satu baris di kode, string tersebut akan menghasilkan tiga baris karena adanya karakter \n.

Input String Menggunakan Scanner

Untuk menerima input teks dari pengguna, Java menggunakan `method nextLine()` dari Scanner.

```
import java.util.Scanner;

class BelajarJava {
  public static void main(String args[]){

    Scanner input = new Scanner(System.in);

    String nama, alamat, hobi;

    System.out.print("Nama: ");
    nama = input.nextLine();

    System.out.print("Alamat: ");
    alamat = input.nextLine();

    System.out.print("Hobi: ");
    hobi = input.nextLine();

    System.out.println("\n## Hasil ##");
    System.out.println("Nama  = " + nama);
    System.out.println("Alamat = " + alamat);
```

```
    System.out.println("Hobi  = " + hobi);
  }
}
```

Pada contoh ini, tiga variabel string diisi dengan input dari pengguna dan kemudian ditampilkan kembali.

String sebagai Objek

Berbeda dengan tipe data primitif, String dalam Java termasuk objek. Artinya, string memiliki method yang dapat dijalankan untuk melakukan berbagai operasi, seperti:

- menghitung jumlah karakter,
- mengubah huruf menjadi besar atau kecil,
- mengambil huruf pada posisi tertentu, dan sebagainya.

Contoh pemanggilan beberapa method bawaan:

```
class BelajarJava {
  public static void main(String args[]){

    String var1 = "Belajar Java di Kampus";

    System.out.println(var1.length());
    System.out.println(var1.toUpperCase());
    System.out.println(var1.toLowerCase());
  }
}
```

Output:

```
24
BELAJAR JAVA DI KAMPUS
belajar java di kampus
```

Mengambil Karakter Tertentu dari String (charAt)

`Method charAt()` digunakan untuk mengambil karakter pada indeks tertentu. Perlu diperhatikan bahwa indeks string dimulai dari 0.

```
class BelajarJava {
  public static void main(String args[]){

    String var1 = "Belajar Java";

    System.out.println(var1.charAt(0));
    System.out.println(var1.charAt(1));
    System.out.println(var1.charAt(2));
  }
}
```

Output:

```
B
e
l
```

Misalnya, `var1.charAt(3)` berarti mengambil karakter keempat dari string.

6) **Tipe Data Array Bahasa Pemrograman Java**

Array adalah struktur data yang digunakan untuk menyimpan sejumlah nilai dengan tipe yang sama dalam satu wadah. Di Indonesia, array juga sering disebut dengan istilah larik. Keberadaan array membantu mengelola banyak data secara lebih praktis dibanding harus membuat variabel satu per satu.

Contoh: untuk menyimpan 5 nilai tanpa array, programmer perlu menulis:

```
int nilai1, nilai2, nilai3, nilai4, nilai5;
```

Ciri-ciri Bilangan Integer di PHP

- Boleh bernilai positif atau negatif.
- Jika tidak diberi tanda +, maka PHP menganggap nilai tersebut sebagai positif.
- Bisa digunakan langsung untuk proses aritmetika seperti +, -, *, atau /.
- PHP otomatis mengenali sebuah variabel sebagai integer jika berisi angka bulat (tanpa deklarasi tipe).

Contoh Penggunaan Integer

```
<?php
$umur = 21;
$harga = 15000;
$rugi  = -500000;

echo $umur;
echo "<br>";
echo $harga;
echo "<br>";
echo $rugi;
?>
```

Integer juga mendukung operasi matematika:

```
<?php
$a = 14;
$b = 16;

$c = $a + $b; // 30
$d = $a * $b; // 224

echo $c . "<br>";
echo $d;
?>
```

Rentang Nilai Integer PHP

Nilai terbesar dan terkecil dari integer tergantung pada arsitektur sistem (biasanya 32-bit). Pada sebagian besar instalasi:

- Minimum: −2,147,483,648
- Maksimum: +2,147,483,647

PHP menyediakan konstanta PHP_INT_MAX untuk mengetahui batas maksimal integer pada sistem yang digunakan:

```
<?php
echo PHP_INT_MAX;
?>
```

Penulisan Integer Berdasarkan Sistem Bilangan

Selain angka desimal (basis 10), PHP juga dapat menyimpan angka dalam format:

- Hexadesimal (basis 16)
- Oktal (basis 8)
- Biner (basis 2)

Format ini jarang digunakan untuk aplikasi web umum, tetapi penting untuk manipulasi data tingkat rendah seperti perhitungan warna (CSS), bitwise, atau setting file permission.

1) Hexadesimal (Base 16)
   Hexadesimal memakai digit: `0−9` dan `A−F`.
   Penulisan di PHP diawali dengan 0x.

```
<?php
$decimal  = 31;
$hexa     = 0x1F; // setara dengan 31
echo $decimal . "<br>";
echo $hexa;
?>
```

2) Oktal (Base 8)
Oktal menggunakan digit 0–7 dan ditulis dengan awalan 0.

```
<?php
$decimal = 511;
$octal   = 0777; // 777 oktal = 511 desimal

echo $decimal . "<br>";
echo $octal;
?>
```

Catatan:
Hindari menulis angka dengan awalan 0 tanpa maksud khusus, karena dapat dianggap sebagai angka oktal.

3) Biner (Base 2)
Biner hanya menggunakan dua digit: 0 dan 1.
Penulisannya memakai awalan 0b.

```
<?php
$decimal = 222;
$binary  = 0b11011110; // sama dengan 222

echo $decimal . "<br>";
echo $binary;
?>
```

b. Tipe Data Float dan Cara Penulisan Float PHP

Tipe data float (atau floating point) digunakan untuk menyimpan angka yang memiliki bagian desimal. Contohnya: `0.9`, `3.14`, `10.25`, dan sebagainya.

Tipe ini dipakai ketika sebuah nilai tidak dapat direpresentasikan sebagai angka bulat, misalnya: nilai IPK mahasiswa, hasil pembagian, perhitungan ilmiah dan angka pecahan lainnya.

Karakteristik Tipe Data Float

- Digunakan untuk angka berpecahan.
- Mendukung nilai positif maupun negatif.
- Umumnya memiliki rentang dari sekitar `1.7 × 10^-308` hingga `1.7 × 10^308`, dengan ketelitian sekitar 15 digit angka.
- Tidak memerlukan deklarasi khusus—PHP otomatis menganggapnya sebagai float jika terdapat tanda desimal.

Cara Penulisan Angka Float di PHP

Dalam PHP (dan mayoritas bahasa pemrograman internasional), pemisah desimal menggunakan tanda titik (.), bukan koma.
Misalnya:

- Penulisan yang benar: 0.87
- Salah dan menyebabkan error: 0,87

PHP menyediakan dua format untuk penulisan angka float:

1) Format Desimal Biasa
   Contoh:
   `0.78, 14.99, 12.5`
2) Scientific Notation (Notasi Ilmiah)
   Digunakan untuk menuliskan angka sangat besar atau sangat kecil secara ringkas.
   Contoh:
   - `0.314E1 → sama dengan 0.314 × 10^1 → 3.14`
   - `12.0E-3 → sama dengan 12.0 × 10^-3 → 0.012`

Contoh Penulisan Float dalam PHP

```
<?php
$angka_float1     = 0.78;
$angka_float2     = 14.99;
$scientific1      = 0.314E1;
$scientific2      = 0.3365E-3;
```

```
echo $angka_float1;     // 0.78
echo "<br>";
echo $angka_float2;     // 14.99
echo "<br>";
echo $scientific1;      // 3.14
echo "<br>";
echo $scientific2;      // 0.0003365
?>
```

Operasi Aritmetika dengan Float

Sama seperti integer, angka float dapat digunakan untuk perhitungan matematika:

```
<?php
$a = 10.66;
$b = 12.4;

$c = $a + $b;
echo $c;  // 23.06

$d = $a / $b;
echo $d;  // 0.85967741935484
?>
```

Seluruh operator dasar seperti +, -, *, / dapat digunakan dengan tipe data float.

c. Tipe Data String dan Cara Penulisan String PHP

String adalah tipe data dalam PHP yang digunakan untuk menyimpan teks, kalimat, atau kumpulan karakter.
Contohnya: `"A"`, `"Belajar PHP"`, `"Hello123?"`.

String sering dipakai untuk menampilkan pesan, memproses input pengguna, maupun mengelola data berbasis teks.

Cara Penulisan String di PHP

PHP menyediakan empat cara utama untuk menuliskan string:

- `Single Quoted (‘ ’)`
- `Double Quoted (“ ”)`
- `Heredoc`
- `Nowdoc`

Masing-masing memiliki aturan dan perilaku yang berbeda.

1) Single Quoted (‘ ’)

   Ini adalah cara penulisan string paling sederhana. Teks ditulis di antara tanda petik satu.

```
<?php
$pesan1 = 'Ini adalah string sederhana';
$pesan2 = 'Dia berkata: "Saya datang."';
$pesan3 = 'Path file: C:\\xampp\\htdocs';
$pesan4 = 'Teks ini tidak memproses \n atau
variabel $pesan1';
?>
```

   Karakter yang tidak diproses:

   - `\n, \t,` dll → dianggap teks biasa
   - variabel ($nama) tidak dievaluasi

   Gunakan `\'` untuk menulis tanda petik satu di dalam string.

2) Double Quoted (“ ”)

   String dengan tanda petik dua dapat memproses karakter khusus dan variabel.

```
<?php
$nama = "Andi";
$kalimat1 = "Halo, $nama";
$kalimat2 = "Pindah baris:\nIni baris baru";
$kalimat3 = "Path: C:\\xampp\\htdocs";
?>
```

Contoh Konversi Boolean

```
<?php
$x = FALSE;       // false
$x = "";          // false
$x = " ";         // true (ada karakter spasi)
$x = 1;           // true
$x = -2;          // true
$x = "belajar";   // true
$x = 3.14;        // true
$x = array();     // false
$x = array(12);   // true
$x = "0";         // false
$x = "false";     // true (string biasa)
?>
```

Catatan penting:

- `""` → `false`
- `" "` → `true`
- `"0"` → `false`
- `"false"` → `true` (karena dianggap string berisi teks)

Kesalahan umum dalam pemrograman terjadi ketika nilai non-boolean ikut dikonversi secara otomatis. Sebaiknya gunakan nilai boolean secara eksplisit untuk mencegah ambigu.

e. Tipe Data Array dan Cara Penulisan Array PHP

Array adalah struktur data yang dapat menyimpan banyak nilai dalam satu variabel. Setiap elemen di dalam array dapat berisi tipe data apa pun: angka, string, boolean, bahkan array lainnya. Penggunaan array membantu mengelompokkan data sejenis dan mengurangi jumlah variabel yang harus dibuat.

Contoh tanpa array:

```
<?php
$nama0 = "Andri";
$nama1 = "Joko";
```

```
$nama2 = "Sukma";
$nama3 = "Rina";
$nama4 = "Sari";
// ... dan seterusnya
?>
```

Pendekatan ini kurang efisien, terutama ketika jumlah data semakin banyak.

Contoh Menggunakan Array

Dengan array, cukup satu variabel untuk menyimpan banyak data:

```
<?php
$nama = array(
    0 => "Andri",
    1 => "Joko",
    2 => "Sukma",
    3 => "Rina",
    4 => "Sari"
);
?>
```

Cara Penulisan Array dalam PHP

1) Array dengan Key dan Value

   PHP menyediakan bentuk standar menggunakan array().

   Format umum:

   ```
   $nama_variabel = array(
       key   => value,
       key2  => value2,
       ...
   );
   ```

   Setiap elemen menyimpan pasangan key (penunjuk posisi) dan value (nilai).

Contoh:

```
<?php
$nama = array(
    1 => "Andri",
    2 => "Joko",
    3 => "Sukma",
    4 => "Rina",
    5 => "Sari"
);

// akses elemen
echo $nama[1]; // Andri
echo "<br>";
echo $nama[2]; // Joko
?>
```

2) Array Tanpa Menentukan Key

Jika key tidak ditulis, PHP otomatis memberi nomor mulai dari 0.

```
<?php
$nama = array("Andri", "Joko", "Sukma",
"Rina", "Sari");

echo $nama[1]; // Joko
echo $nama[2]; // Sukma
?>
```

Penting: indeks array PHP selalu dimulai dari 0, bukan 1.

3) Array dengan Beragam Jenis Key dan Value

PHP memperbolehkan key bertipe integer, string, maupun boolean. Value juga boleh berupa berbagai tipe data.

```
<?php
$coba = array(
    2     => "Andri",
    "dua" => "2",
```

```
    'tiga'=> 3,
    true  => true,
    9     => "sembilan"
);

echo $coba[2];     // Andri
echo $coba["dua"]; // 2
echo $coba['tiga']; // 3
echo  $coba[true];     // 1 (true dikonversi
menjadi 1)
echo $coba[9];      // sembilan
?>
```

Catatan: Jika key menggunakan boolean, PHP otomatis mengubah:

- true → 1
- false → 0

4) Short Syntax Array (PHP 5.4+)

PHP menyediakan cara singkat menggunakan tanda [].

```
<?php
$nama = ["Andri", "Joko", "Sukma", "Rina",
"Sari"];

echo $nama[1]; // Joko
echo "<br>";
echo $nama[3]; // Rina
?>
```

Bentuk ini lebih umum digunakan pada PHP modern.

E. Operasi String

String adalah rangkaian karakter yang digunakan untuk menyimpan teks. Hampir semua bahasa pemrograman menyediakan berbagai operasi dasar seperti penggabungan, pemotongan, dan pencarian. Walaupun konsepnya sama, cara penulisannya dapat berbeda untuk setiap bahasa.

1. **Penggabungan String (Concatenation)**
   Penggabungan string dilakukan dengan menghubungkan dua atau lebih teks menjadi satu kesatuan.
   Python
   ```
   nama = "Andi"
   sapaan = "Halo " + nama
   ```
   C++
   ```
   string nama = "Andi";
   string sapaan = "Halo " + nama;
   ```
   Java
   ```
   String nama = "Andi";
   String sapaan = "Halo " + nama;
   ```
   PHP
   ```
   $nama = "Andi";
   $sapaan = "Halo " . $nama;
   ```

2. **Pemotongan String (Substring / Slicing)**
   Pemotongan string digunakan untuk mengambil sebagian teks berdasarkan posisi karakter.
   Python
   ```
   teks = "Pemrograman"
   bagian = teks[0:3]   # 'Pem'
   ```
   C++
   ```
   string teks = "Pemrograman";
   string bagian = teks.substr(0, 3);
   ```

Java

```
String teks = "Pemrograman";
String bagian = teks.substring(0, 3);
```

PHP

```
$teks = "Pemrograman";
$bagian = substr($teks, 0, 3);
```

3. **Pencarian String**

   Pencarian string digunakan untuk mengetahui apakah sebuah karakter/kata berada di dalam string tertentu.

   Python

   ```
   "pro" in "Pemrograman"   # True
   ```

   C++

   ```
   teks.find("pro") != string::npos;
   ```

   Java

   ```
   teks.contains("pro");
   ```

   PHP

   ```
   strpos("Pemrograman", "pro") !== false;
   ```

F. Struktur Data Sederhana: List dan Tuple

Struktur data adalah cara menyimpan dan mengelola kumpulan nilai. Pada tahap dasar, dua struktur yang paling sering ditemui adalah **list** dan **tuple**. Tidak semua bahasa memiliki istilah yang sama, tetapi konsepnya bisa dijelaskan sebagai berikut:

1. **List (Koleksi yang Dapat Diubah)**

   List adalah struktur data yang bersifat *mutable*, artinya isinya dapat ditambah, dihapus, atau diubah.

Python (List Asli)

```
angka = [1, 2, 3]
angka[1] = 10
```

C++ (Vector sebagai padanan List)

```
vector<int> angka = {1, 2, 3};
angka[1] = 10;
```

Java (ArrayList sebagai padanan List)

```
ArrayList<Integer> angka = new ArrayList<>();
angka.add(1);
angka.add(2);
angka.add(3);
angka.set(1, 10);
```

PHP (Array sebagai List)

```
$angka = [1, 2, 3];
   $angka[1] = 10;
```

2. Tuple (Koleksi yang Tidak Diubah)

Tuple adalah struktur data *immutable*, artinya isi di dalamnya tidak dapat dimodifikasi setelah dibuat.

Catatan: Tidak semua bahasa memiliki tuple secara langsung. C++ dan Java menggunakan alternatif seperti `pair` atau `final array`.

Python

```
data = (1, 2, 3)
```

C++ (Menggunakan std::tuple)

```
tuple<int, int, int> data = make_tuple(1, 2, 3);
```

Java (Tidak Ada Tuple Bawaan, Solusi: record/class atau array final)

```
final int[] data = {1, 2, 3};
```

Tabel 4. 1 Daftar Operator Aritmatika

Operator	Fungsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa hasil bagi (modulus)

Contoh Program Operator Aritmatika

1. **Python**

```
# Operator Aritmatika dalam Python

a = 10
b = 3

print("Penjumlahan:", a + b)
print("Pengurangan:", a - b)
print("Perkalian:", a * b)
print("Pembagian:", a / b)
print("Modulus:", a % b)
```

2. **C++**

```
#include <iostream>
using namespace std;

int main() {
    int a = 10, b = 3;

    cout << "Penjumlahan: " << a + b << endl;
    cout << "Pengurangan: " << a - b << endl;
    cout << "Perkalian: " << a * b << endl;
    cout << "Pembagian: " << a / b << endl;
    cout << "Modulus: " << a % b << endl;

    return 0;
}
```

3. Java

```
public class OperatorAritmatika {
    public static void main(String[] args) {
        int a = 10, b = 3;

        System.out.println("Penjumlahan: " + (a + b));
        System.out.println("Pengurangan: " + (a - b));
        System.out.println("Perkalian: " + (a * b));
        System.out.println("Pembagian: " + (a / b));
        System.out.println("Modulus: " + (a % b));
    }
}
```

4. PHP

```
<?php
$a = 10;
$b = 3;

echo "Penjumlahan: " . ($a + $b) . "<br>";
echo "Pengurangan: " . ($a - $b) . "<br>";
echo "Perkalian: " . ($a * $b) . "<br>";
echo "Pembagian: " . ($a / $b) . "<br>";
echo "Modulus: " . ($a % $b) . "<br>";
?>
```

C. Operator Perbandingan

Operator perbandingan digunakan untuk mengevaluasi dua nilai dan menentukan hubungan di antara keduanya, seperti apakah nilai tersebut sama, tidak sama, lebih besar, lebih kecil, atau memiliki kondisi tertentu yang saling berkaitan. Setiap operasi perbandingan selalu menghasilkan output berupa nilai Boolean, yaitu true jika perbandingan tersebut benar sesuai kondisi yang diuji, dan false jika tidak. Operator ini sangat penting dalam proses pengambilan keputusan dalam program, karena sering digunakan dalam struktur kontrol seperti if, while, dan for untuk menentukan alur logika berdasarkan kondisi yang diberikan.

3. Java

```
public class OperatorLogika {
    public static void main(String[] args) {
        boolean x = true, y = false;

        System.out.println(x && y);
        System.out.println(x || y);
        System.out.println(!x);
    }
}
```

4. PHP

```
<?php
$x = true;
$y = false;

var_dump($x && $y);
var_dump($x || $y);
var_dump(!$x);
?>
```

E. Operator Boolean

Operator Boolean digunakan untuk mengolah nilai benar (*true*) atau salah (*false*). Secara praktik, operator Boolean sering kali merupakan penggunaan kompleks dari operator logika dan perbandingan.
Contohnya:

- Menguji beberapa kondisi dalam satu ekspresi
- Menggunakan hasil perbandingan sebagai syarat keputusan

Contoh Program Operator Boolean

1. **Python**

```
a = 7
b = 3

hasil = (a > b) and (b < 10)
print(hasil)
```

2. **C++**

```
#include <iostream>
using namespace std;

int main() {
    int a = 7, b = 3;

    bool hasil = (a > b) && (b < 10);
    cout << hasil << endl;

    return 0;
}
```

3. **Java**

```
public class OperatorBoolean {
    public static void main(String[] args) {
        int a = 7, b = 3;

        boolean hasil = (a > b) && (b < 10);
        System.out.println(hasil);
    }
}
```

4. **PHP**

```
<?php
$a = 7;
$b = 3;

$hasil = ($a > $b) && ($b < 10);
var_dump($hasil);
?>
```

F. Operator String

Operator string digunakan untuk memanipulasi teks. Operasi paling umum adalah penggabungan (concatenation) dan pengulangan (khusus Python).

Contoh Program Operator String

1. **Python —** (+) untuk penggabungan

```
nama = "Buku"
jenis = "Pemrograman"
hasil = nama + " " + jenis
print(hasil)
```

2. **C++ — operator (+) tidak ada, gunakan "append" atau "<<"**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string nama = "Buku";
    string jenis = "Pemrograman";
    string hasil = nama + " " + jenis;

    cout << hasil;
    return 0;
}
```

3. **Java — (+) untuk concatenation**

```
public class StringOp {
    public static void main(String[] args) {
        String nama = "Buku";
        String jenis = "Pemrograman";
        String hasil = nama + " " + jenis;

        System.out.println(hasil);
    }
}
```

4. **PHP — (.) untuk concatenation**

```
<?php
$nama = "Buku";
$jenis = "Pemrograman";
$hasil = $nama . " " . $jenis;

echo $hasil;
?>
```

G. Operator Increment & Decrement

Operator ini digunakan untuk menambah atau mengurangi nilai variabel sebesar 1 secara otomatis tanpa harus menuliskan operasi penjumlahan atau pengurangan secara lengkap. Terdapat empat bentuk utama, yaitu:

- **++a (pre-increment)** → nilai ditambah 1 terlebih dahulu, lalu digunakan dalam ekspresi.
- **a++ (post-increment)** → nilai digunakan dalam ekspresi terlebih dahulu, baru kemudian ditambah 1.
- **--a (pre-decrement)** → nilai dikurangi 1 terlebih dahulu sebelum digunakan.
- **a-- (post-decrement)** → nilai digunakan dulu, kemudian dikurangi 1.

Python tidak memiliki operator ++ / --, sehingga hanya memakai operasi += 1 dan -= 1.

Contoh Program Operator Boolean

1. **Python**

```
a = 5
a += 1    # increment
a -= 1    # decrement
print(a)
```

2. **C++**

```
#include <iostream>
using namespace std;

int main() {
    int a = 5;
    cout << ++a << endl; // pre-increment
    cout << a++ << endl; // post-increment
    cout << a << endl;

    return 0;
}
```

3. **Java**

```
public class IncDec {
    public static void main(String[] args) {
        int a = 5;
        System.out.println(++a);
        System.out.println(a++);
        System.out.println(a);
    }
}
```

4. **PHP**

```
<?php
$a = 5;
echo ++$a . "\n";
echo $a++ . "\n";
echo $a . "\n";
?>
```

H. Operator Bitwise

Operator bitwise bekerja pada level bit, digunakan dalam operasi yang memerlukan manipulasi biner.

Tabel 4. 4 Daftar Operator Aritmatika

Operator	Arti
&	AND
|	OR
^	XOR
~	NOT
<<	shift kiri
>>	shift kanan

Contoh Program Operator Bitwise

1. **Python**

```
a = 6   # 110
b = 3   # 011

print(a & b)
print(a | b)
print(a ^ b)
print(~a)
print(a << 1)
print(a >> 1)
```

2. **C++**

```
#include <iostream>
using namespace std;

int main() {
    int a = 6, b = 3;

    cout << (a & b) << endl;
    cout << (a | b) << endl;
    cout << (a ^ b) << endl;
```

```
    cout << (~a) << endl;
    cout << (a << 1) << endl;
    cout << (a >> 1) << endl;

    return 0;
}
```

3. Java

```
#include <iostream>
using namespace std;

int main() {
    int a = 6, b = 3;

    cout << (a & b) << endl;
    cout << (a | b) << endl;
    cout << (a ^ b) << endl;
    cout << (~a) << endl;
    cout << (a << 1) << endl;
    cout << (a >> 1) << endl;

    return 0;
}
```

4. PHP

```
<?php
$a = 6;
$b = 3;

echo ($a & $b) . "\n";
echo ($a | $b) . "\n";
echo ($a ^ $b) . "\n";
echo (~$a) . "\n";
echo ($a << 1) . "\n";
echo ($a >> 1) . "\n";
?>
```

```
    cout << "p1 == p3 : " << (p1 == p3) << endl; //
False (alamat berbeda)

    // KEANGGOTAAN (menggunakan find)
    vector<int> data = {1, 2, 3};

    cout << "\nKEANGGOTAAN:\n";
    cout << "2 in data? "
         << (find(data.begin(), data.end(), 2) !=
data.end())
         << endl;

    cout << "5 in data? "
         << (find(data.begin(), data.end(), 5) !=
data.end())
         << endl;

    return 0;
}
```

Penjelasan Kode C++ (Point Form)

- Operator Identitas (Menggunakan Alamat Memori)
  Karena C++ tidak memiliki operator `is` seperti Python, identitas dicek melalui alamat memori menggunakan pointer.
  - `int x = 10;` dan `int y = 10;`
    Dua variabel berbeda meskipun nilainya sama.
  - `p1 = &x` dan `p2 = &x`
    Kedua pointer menunjuk alamat memori yang sama → identitas sama.
  - `p3 = &y`
    Pointer menunjuk alamat memori berbeda dari `x`.

  Hasil:
  - `p1 == p2` → `True`, karena alamat memori sama.
  - `p1 == p3` → `False`, karena menunjuk ke variabel yang berbeda.

- Operator Keanggotaan (Menggunakan `find()`)
  C++ tidak memiliki operator `in`, sehingga pemeriksaan dilakukan menggunakan algoritma `find()` dari `<algorithm>`.
  - `vector<int> data = {1, 2, 3};`
  - `find(data.begin(), data.end(), 2)` mencari angka `2` dalam vector.
  - Jika hasil `find` tidak sama dengan `data.end()`, artinya elemen ditemukan.

  Hasil:
  - `2 in data` → `True`, karena 2 ada dalam vector.
  - `5 in data` → `False`, karena 5 tidak ditemukan.

3. Java — Identitas & Keanggotaan Versi Java

Java membedakan identitas objek dan kesetaraan nilai:

- Identitas: menggunakan == → membandingkan referensi objek
- Identitas: menggunakan `.equals()` → membandingkan nilai
- Keanggotaan: menggunakan metode `.contains()` pada List.

Contoh Program Java Lengkap

```
import java.util.*;

public class IdentityMembership {
    public static void main(String[] args) {

        // IDENTITAS
        String a = new String("halo");
        String b = a;
        String c = new String("halo");

        System.out.println("IDENTITAS:");
        System.out.println("a == b : " + (a == b));
// True
        System.out.println("a == c : " + (a == c));
// False
```

```
        System.out.println("a.equals(c) : " +
a.equals(c)); // True (nilai sama)

        // KEANGGOTAAN
        List<Integer> angka = Arrays.asList(1, 2, 3);

        System.out.println("\nKEANGGOTAAN:");
        System.out.println("angka.contains(2): " +
angka.contains(2));
        System.out.println("angka.contains(5): " +
angka.contains(5));
    }
}
```

Penjelasan Kode Java (Point Form)

- Operator Identitas (== untuk referensi objek)

  Di Java, operator == tidak membandingkan isi, tetapi membandingkan referensi objek, yaitu apakah dua variabel menunjuk objek yang sama di memori.

  - `String a = new String("halo");`
    Membuat objek String baru.
  - `String b = a;`
    `b` menunjuk objek yang sama dengan `a`.
  - `String c = new String("halo");`
    Membuat objek baru dengan isi yang sama, tetapi referensi berbeda.

  Hasil:

  - `a == b` → `True`, karena kedua variabel menunjuk objek yang sama.
  - `a == c` → `False`, karena referensi berbeda meskipun nilainya sama.
  - `a.equals(c)` → `True`, karena `equals()` membandingkan isi teks, bukan identitas objek.

- Operator Keanggotaan (`contains()`)
  Java tidak memiliki operator `in` seperti Python, tetapi menyediakan metode:
  - `List.contains(element)`
  - yang mengembalikan `true` jika nilai tersebut ada dalam list.
  - `List<Integer> angka = Arrays.asList(1, 2, 3);`

  Hasil:
  - `angka.contains(2)` → `True`, karena 2 ada dalam list.
  - `angka.contains(5)` → `False`, karena 5 tidak ada dalam list.

4. PHP — Identitas & Keanggotaan dalam PHP

PHP menggunakan:

- `===` → identitas (tipe dan nilai harus sama)
- `in_array()` → keanggotaan
- `!==` → kebalikannya

Contoh Program PHP Lengkap

```
<?php
// IDENTITAS
$a = 5;
$b = "5";

echo "IDENTITAS:\n";
var_dump($a === $b); // false (tipe berbeda)
var_dump($a === 5);  // true (tipe dan nilai sama)

// KEANGGOTAAN
$data = [1, 2, 3];

echo "\nKEANGGOTAAN:\n";
var_dump(in_array(2, $data));  // true
var_dump(in_array(5, $data));  // false
```

Penjelasan Kode PHP (Point Form)

- Operator Identitas (===)
  Di PHP, operator identitas === digunakan untuk membandingkan nilai dan tipe data secara sekaligus.
  - `$a = 5;` → `integer`
  - `$b = "5";` → `string`

  Hasil:
  - `$a === $b` → `false`, karena meskipun nilainya sama, tipe datanya berbeda (integer vs string).
  - `$a === 5` → `true`, karena nilai dan tipe datanya sama.

- Operator Keanggotaan (in_array())
  PHP tidak memiliki operator in, tetapi menyediakan fungsi:
  - `in_array(value, array)`
  - yang mengembalikan true jika nilai ada dalam array.
  - `$data = [1, 2, 3];`

  Hasil:
  - `in_array(2, $data)` → `true`, karena 2 ada dalam array.
  - `in_array(5, $data)` → `false`, karena 5 tidak ditemukan.

K. Operator Gabungan Assignment

Operator ini menggabungkan operasi aritmatika atau logika dengan proses assignment, sehingga memungkinkan suatu variabel menjalani perhitungan sekaligus diperbarui nilainya dalam satu langkah. Dengan menggunakan operator ini, kode menjadi lebih ringkas dan efisien karena tidak perlu menuliskan operasi secara panjang, misalnya `a = a + 5` dapat disederhanakan menjadi `a += 5`, tanpa mengubah hasil akhirnya.

Tabel 4. 5 Daftar Operator Aritmatika

Operator	Arti
+=	tambah lalu simpan
-=	kurang lalu simpan
*=	kali lalu simpan
/=	bagi lalu simpan
%=	modulus lalu simpan
**=	pangkat (Python)
<<= / >>=	shift bit

Contoh Program

1. **Python**

```
a = 10
a += 5
a *= 2
print(a)
```

2. **C++**

```
#include <iostream>
using namespace std;

int main() {
    int a = 10;
    a += 5;
    a *= 2;
    cout << a;
    return 0;
}
```

3. **Java**

```
public class AssignCombo {
    public static void main(String[] args) {
        int a = 10;
        a += 5;
        a *= 2;
        System.out.println(a);
    }
}
```

C. Jenis-Jenis Output

Output adalah cara program menyampaikan hasil proses atau informasi kepada pengguna, sistem lain, atau perangkat eksternal (Dafit Afianto & Riko Muhammad Suri, 2025). Output memastikan program tidak hanya bekerja secara internal, tetapi juga bermanfaat bagi pengguna dan lingkungan sekitarnya. Berdasarkan media atau cara penyampaiannya, output dapat dibagi menjadi beberapa jenis:

1. Output ke Layar

Output ke layar menampilkan hasil perhitungan, pesan, atau data penting langsung kepada pengguna. Contoh: Program kalkulator yang menampilkan hasil penjumlahan, atau aplikasi ujian yang menampilkan nilai siswa. Layar menjadi sarana komunikasi utama antara program dan manusia, memungkinkan pengguna melihat hasil proses secara langsung, menilai, dan mengambil keputusan berdasarkan informasi yang ditampilkan. Output ini dapat berupa teks, grafik, atau antarmuka interaktif.

2. Output ke File

Output ke file digunakan untuk menyimpan hasil proses secara permanen agar dapat digunakan atau diakses di kemudian hari. File dapat berupa TXT, CSV, JSON, atau format lain sesuai kebutuhan aplikasi. Contoh: Laporan bulanan, log aktivitas, atau catatan transaksi disimpan agar dapat dianalisis lebih lanjut atau dijadikan arsip. Output ke file memastikan data tersimpan dengan aman dan dapat digunakan kembali tanpa harus memasukkan informasi secara manual.

3. Output ke Perangkat Eksternal

Beberapa program perlu mengirim informasi ke perangkat keras di luar komputer, seperti printer, layar LCD, atau speaker. Contoh: Mesin kasir mencetak struk pembelian, papan informasi digital menampilkan jadwal atau pesan, dan sistem audio memutar suara peringatan. Output ke

perangkat eksternal memungkinkan program memberikan efek fisik atau respons nyata di lingkungan sekitar.

4. Output ke Jaringan

Output ke jaringan memungkinkan program mengirim data ke sistem lain melalui internet atau intranet, termasuk server, API, atau database. Contoh: Aplikasi perbankan online yang mengirim data transaksi secara real-time ke server pusat, atau sistem inventaris berbasis cloud yang menyinkronkan stok barang antar cabang. Output jenis ini penting dalam aplikasi modern yang membutuhkan konektivitas, kolaborasi antar pengguna, dan akses data secara global.

D. Sintaks Input dan Output

Input digunakan ketika program membutuhkan data yang dimasukkan langsung oleh pengguna. Walaupun tujuannya sama, cara penulisan sintaks input berbeda di setiap bahasa. Sedangkan Output digunakan untuk menampilkan teks, angka, atau hasil perhitungan ke layar. Setiap bahasa menawarkan fungsi atau perintah untuk mencetak informasi.

1. Python

```
# Input
nama = input("Masukkan nama: ")
usia = int(input("Masukkan usia: "))

# Output
print("Halo", nama)
print("Usia Anda:", usia)
```

Di Python, `input()` selalu mengembalikan string. Jika ingin menerima angka, kita harus mengonversinya dengan `int()` atau `float()`.

2. C++

```
#include <iostream>
using namespace std;
```

```
int main() {
    string nama;
    int usia;

    cout << "Masukkan nama: ";
    cin >> nama;
    cout << "Masukkan usia: ";
    cin >> usia;

    cout << "Halo " << nama << endl;
    cout << "Usia Anda: " << usia << endl;

    return 0;
}
```

C++ menggunakan cin untuk input dan cout untuk output. cin membaca input sampai spasi, sehingga untuk input string panjang bisa menggunakan `getline()`.

3. **Java**

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Masukkan nama: ");
        String nama = sc.nextLine();

        System.out.print("Masukkan usia: ");
        int usia = sc.nextInt();

        System.out.println("Halo " + nama);
        System.out.println("Usia Anda: " + usia);
    }
}
```

Java menggunakan Scanner untuk membaca input. Fungsi `nextLine()` untuk string dan `nextInt()` untuk integer.

4. PHP

```
<?php
$nama = readline("Masukkan nama: ");
$usia = intval(readline("Masukkan usia: "));

echo "Halo $nama\n";
echo "Usia Anda: $usia\n";
?>
```

PHP menggunakan `readline()` untuk input dari console dan echo atau print untuk output. Konversi tipe data dapat dilakukan dengan `intval()`, `floatval()`, dll.

E. Format Output

Format output adalah metode yang digunakan program untuk menampilkan data sehingga mudah dibaca dan sesuai dengan kebutuhan pengguna (Richki Hardi, 2021). Dengan format output, program dapat menyajikan informasi secara rapi, terstruktur, dan konsisten, baik berupa teks, angka, maupun kombinasi keduanya. Hal ini sangat penting agar pengguna dapat memahami hasil proses program dengan cepat dan tepat.

Dalam berbagai bahasa pemrograman seperti Python, C++, Java, dan PHP, tersedia cara-cara khusus untuk menampilkan output dengan format tertentu. Misalnya, di Python ada `f-string` yang memungkinkan nilai variabel langsung disisipkan ke dalam teks, sedangkan di C++ dan Java terdapat `printf-style` yang menggunakan simbol khusus untuk menentukan tipe data dan posisi nilai. Di PHP, printf juga digunakan untuk tujuan yang sama. Metode-metode ini membantu programmer menyusun output agar informasi tersaji dengan format yang jelas dan terkontrol.

Selain itu, banyak bahasa pemrograman memungkinkan pengaturan tambahan seperti jumlah desimal, lebar kolom, atau kombinasi string dan angka. Format output seperti ini mempermudah penyajian data

dengan presisi tinggi, membuat laporan lebih profesional, dan meningkatkan keterbacaan antarmuka atau layar program. Dengan demikian, penggunaan format output tidak hanya mempercantik tampilan, tetapi juga meningkatkan akurasi dan efektivitas penyampaian informasi di berbagai aplikasi, baik berbasis desktop, web, maupun sistem terintegrasi.

Tabel 5. 1 Cara Menampilkan Output dengan Format

Bahasa	**Contoh Format**
Python	`f"Nama: {nama}, Usia: {usia}"`
C++	`printf("Nama: %s, Usia: %d", nama, usia);`
Java	`System.out.printf("Nama: %s, Usia: %d", nama, usia);`
PHP	`printf("Nama: %s, Usia: %d", $nama, $usia);`

Contoh Program Format Output

1. **Python (menggunakan f-string)**

```
# Program menampilkan nama dan nilai siswa
nama = "Andi"
nilai = 85

# Menggunakan f-string untuk format output
print(f"Nama siswa: {nama}, Nilai: {nilai}")
```

Output:

```
Nama siswa: Andi, Nilai: 85
```

2. **C++ (menggunakan printf-style dan iostream)**

```
#include <iostream>
#include <cstdio>
using namespace std;

int main() {
    string nama = "Andi";
```

```
    int nilai = 85;

    // Menggunakan cout (iostream)
    cout << "Nama siswa: " << nama << ", Nilai: "
<< nilai << endl;

    // Menggunakan printf (printf-style)
    printf("Nama   siswa:   %s,   Nilai:   %d\n",
nama.c_str(), nilai);

    return 0;
}
```

Output:

```
Nama siswa: Andi, Nilai: 85
Nama siswa: Andi, Nilai: 85
```

3. **Java (menggunakan System.out.printf)**

```
public class Main {
    public static void main(String[] args) {
        String nama = "Andi";
        int nilai = 85;

        // Menggunakan println
        System.out.println("Nama siswa: " + nama
+ ", Nilai: " + nilai);

        // Menggunakan printf (format output)
        System.out.printf("Nama siswa: %s, Nilai:
%d\n", nama, nilai);
    }
}
```

Output:

```
Nama siswa: Andi, Nilai: 85
Nama siswa: Andi, Nilai: 85
```

4. **PHP (menggunakan printf dan echo)**

```
<?php
$nama = "Andi";
$nilai = 85;

// Menggunakan echo
echo "Nama siswa: $nama, Nilai: $nilai\n";

// Menggunakan printf (format output)
printf("Nama siswa: %s, Nilai: %d\n", $nama,
$nilai);
?>
```

Output:

```
Nama siswa: Andi, Nilai: 85
Nama siswa: Andi, Nilai: 85
```

F. Validasi Input

Dalam pemrograman, input dari pengguna adalah sumber data utama yang akan diproses oleh program. Namun, data yang masuk tidak selalu sesuai dengan harapan program. Oleh karena itu, validasi input menjadi sangat penting untuk memastikan program berjalan aman, stabil, dan menghasilkan output yang benar.

Validasi input bertujuan untuk:

- Mencegah error program: Input yang salah dapat menyebabkan crash atau exception.
- Menjamin data konsisten: Memastikan data sesuai tipe, format, atau rentang yang diinginkan.
- Meningkatkan keamanan: Mencegah input berbahaya seperti script injection atau SQL injection.

Contoh IF-ELSE:

Python

```
angka = 7
if angka % 2 == 0:
    print("Bilangan genap")
else:
    print("Bilangan ganjil")
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int angka = 7;
    if (angka % 2 == 0) {
        cout << "Bilangan genap" << endl;
    } else {
        cout << "Bilangan ganjil" << endl;
    }
    return 0;
}
```

Java

```
public class Main {
    public static void main(String[] args) {
        int angka = 7;
        if (angka % 2 == 0) {
            System.out.println("Bilangan genap");
        } else {
            System.out.println("Bilangan ganjil");
        }
    }
}
```

PHP

```
<?php
$angka = 7;
if ($angka % 2 == 0) {
    echo "Bilangan genap";
} else {
    echo "Bilangan ganjil";
}
?>
```

D. IF-ELIF-ELSE (atau IF-ELSE IF-ELSE)

Ketika ada lebih dari dua kemungkinan hasil dari suatu kondisi, IF-ELIF-ELSE menjadi pilihan yang tepat. Struktur ini memungkinkan program memeriksa beberapa kondisi secara berurutan, dan mengeksekusi blok kode yang pertama kali memenuhi kondisi. Jika tidak ada kondisi yang terpenuhi, blok ELSE (opsional) dapat digunakan sebagai tindakan default.

Misalnya, dalam sistem penilaian akademik: jika nilai ≥ 85, siswa mendapat predikat "A"; jika nilai ≥ 70, predikat "B"; jika nilai ≥ 55, predikat "C"; dan jika nilai di bawah 55, predikat "D". Dengan IF-ELIF-ELSE, kita dapat mengeksekusi blok kode yang tepat tanpa menulis banyak IF tunggal yang terpisah, sehingga kode menjadi lebih rapi dan mudah dipelihara.

Contoh IF-ELIF-ELSE:

Python

```
umur = 20
if umur < 13:
    print("Anak-anak")
elif umur < 20:
    print("Remaja")
else:
```

```
    print("Dewasa")
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int umur = 20;
    if (umur < 13) {
        cout << "Anak-anak" << endl;
    } else if (umur < 20) {
        cout << "Remaja" << endl;
    } else {
        cout << "Dewasa" << endl;
    }
    return 0;
}
```

Java

```
public class Main {
    public static void main(String[] args) {
        int umur = 20;
        if (umur < 13) {
            System.out.println("Anak-anak");
        } else if (umur < 20) {
            System.out.println("Remaja");
        } else {
            System.out.println("Dewasa");
        }
    }
}
```

PHP

```
<?php
$umur = 20;
```

```
if ($umur < 13) {
    echo "Anak-anak";
} else if ($umur < 20) {
    echo "Remaja";
} else {
    echo "Dewasa";
}
?>
```

E. SWITCH / CASE

Switch digunakan ketika program harus memilih satu dari beberapa kemungkinan blok kode berdasarkan nilai sebuah variabel. Switch cocok digunakan ketika terdapat banyak kondisi tetap yang harus dibandingkan dengan nilai yang sama, sehingga lebih ringkas dibanding menulis banyak IF-ELSE IF-ELSE.

Di beberapa bahasa pemrograman:

- C++, Java, PHP: switch tersedia secara bawaan. Program memeriksa nilai variabel dan mengeksekusi blok case yang sesuai. Jika tidak ada yang cocok, blok default akan dijalankan.
- Python: Python tidak memiliki switch bawaan, namun dapat menggunakan dictionary atau match-case (Python 3.10 ke atas) untuk meniru fungsionalitas switch.

Switch/Case membuat kode lebih mudah dibaca dan lebih terstruktur, terutama ketika menangani banyak kemungkinan nilai yang berbeda dari sebuah variabel, misalnya menu pilihan, kode status, atau kategori tertentu.

Contoh SWITCH:

Python

```
hari = 3
```

```
hari_dict = {1: "Senin", 2: "Selasa", 3: "Rabu"}
print(hari_dict.get(hari, "Hari tidak diketahui"))
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int hari = 3;
    switch(hari) {
        case 1: cout << "Senin" << endl; break;
        case 2: cout << "Selasa" << endl; break;
        case 3: cout << "Rabu" << endl; break;
        default: cout << "Hari tidak diketahui" <<
endl;
    }
    return 0;
}
```

Java

```
public class Main {
    public static void main(String[] args) {
        int hari = 3;
        switch(hari) {
            case 1: System.out.println("Senin");
break;
            case 2: System.out.println("Selasa");
break;
            case 3: System.out.println("Rabu");
break;
            default: System.out.println("Hari tidak
diketahui");
        }
    }
}
```

Contoh: meminta input angka positif

Python:

```
while True:
    angka = int(input("Masukkan angka positif: "))
    if angka > 0:
        break
print("Angka diterima:", angka)
```

C++:

```
#include <iostream>
using namespace std;

int main() {
    int angka;
    do {
        cout << "Masukkan angka positif: ";
        cin >> angka;
    } while(angka <= 0);
    cout << "Angka diterima: " << angka << endl;
    return 0;
}
```

Java:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int angka;
        do {
            System.out.print("Masukkan angka positif:
");
            angka = sc.nextInt();
        } while(angka <= 0);
        System.out.println("Angka diterima: " +
angka);
    }
}
```

PHP:
```
<?php
do {
    $angka = (int)readline("Masukkan angka positif:
");
} while($angka <= 0);
echo "Angka diterima: $angka\n";
?>
```

E. Infinite Loop

Infinite loop adalah perulangan yang berjalan tanpa henti hingga dihentikan secara eksplisit menggunakan break atau kondisi lain. Infinite loop sering digunakan dalam program server atau program real-time.

Contoh: mencetak pesan sekali saja (dihentikan dengan break)
Python:
```
while True:
    print("Looping tak terbatas")
    break
```

C++:
```
#include <iostream>
using namespace std;

int main() {
    while(true){
        cout << "Looping tak terbatas" << endl;
        break;
    }
    return 0;
}
```

Java:

```
public class Main {
    public static void main(String[] args) {
        for(;;){
            System.out.println("Looping tak
terbatas");
            break;
        }
    }
}
```

PHP:

```
<?php
while(true){
    echo "Looping tak terbatas\n";
    break;
}
?>
```

F. Kendali Looping: Break, Continue, Pass

Dalam struktur perulangan, sering kali diperlukan kendali khusus untuk menghentikan, melewati, atau menandai bagian tertentu dari loop. Tiga perintah yang umum digunakan untuk mengendalikan alur looping adalah break, continue, dan pass (khusus Python). Setiap perintah memiliki fungsi yang berbeda dan membantu programmer mengatur logika program dengan lebih fleksibel.

Break digunakan untuk menghentikan loop secara langsung, meskipun kondisi perulangan belum selesai terpenuhi. Ketika perintah ini dijalankan, program akan keluar dari loop dan melanjutkan eksekusi ke baris setelah loop. Perintah ini berguna ketika ditemukan kondisi khusus yang membuat proses perulangan tidak perlu dilanjutkan lagi, seperti ketika data yang dicari sudah ditemukan.

Continue berfungsi untuk melewatkan iterasi saat ini dan langsung melanjutkan ke iterasi berikutnya. Program tidak keluar dari loop, tetapi hanya mengabaikan sisa perintah pada putaran tersebut. Perintah ini sangat membantu ketika ingin menghindari kasus tertentu, misalnya melewati data yang tidak valid atau mengabaikan nilai tertentu tanpa menghentikan seluruh proses perulangan.

Sementara itu, Pass adalah perintah khusus dalam Python yang tidak melakukan apa-apa ketika dijalankan. Meskipun terlihat sederhana, pass sangat berguna sebagai placeholder untuk menyusun kerangka program sebelum logika lengkapnya ditambahkan. Programmer sering menggunakannya ketika ingin menunda penulisan isi blok kode namun tetap menjaga struktur program tetap valid secara sintaks.

Sebagai contoh, ketika memproses daftar angka, perintah continue dapat digunakan untuk melewati angka negatif, sehingga hanya angka positif yang diproses oleh program. Dengan memanfaatkan break, continue, dan pass secara tepat, logika kontrol dalam loop dapat diatur dengan lebih efektif dan sesuai kebutuhan program.

Contoh: melewati angka negatif dalam daftar

Python:

```
angka_list = [3, -1, 5, -2, 7]
for a in angka_list:
    if a < 0:
        continue
    print(a)
```

C++:

```
#include <iostream>
using namespace std;

int main() {
    int angka_list[] = {3, -1, 5, -2, 7};
    for(int i = 0; i < 5; i++){
        if(angka_list[i] < 0) continue;
        cout << angka_list[i] << endl;
```

```
    }
    return 0;
}
```

Java:

```
public class Main {
    public static void main(String[] args) {
        int[] angka_list = {3, -1, 5, -2, 7};
        for(int a : angka_list){
            if(a < 0) continue;
            System.out.println(a);
        }
    }
}
```

PHP:

```
<?php
$angka_list = [3, -1, 5, -2, 7];
foreach($angka_list as $a){
    if($a < 0) continue;
    echo $a . "\n";
}
?>
```

G. Studi Kasus: Pencarian Data dalam List/Array

Pencarian data adalah operasi dasar dalam pemrograman, yang memungkinkan program menemukan nilai tertentu dalam sebuah struktur data, seperti list (Python), array (C++, Java, PHP). Pada contoh ini, kita akan meminta pengguna memasukkan angka yang ingin dicari, kemudian program akan menelusuri semua elemen untuk mengetahui apakah angka tersebut ada dalam data.

Python

```
# Membuat daftar data
data = [10, 20, 30, 40, 50]

# Membaca input dari pengguna
cari = int(input("Masukkan angka yang dicari: "))

# Inisialisasi variabel ditemukan
ditemukan = False

# Perulangan untuk mencari data
for d in data:
    if d == cari:
        ditemukan = True
        break

# Menampilkan hasil
if ditemukan:
    print(f"Angka {cari} ditemukan di daftar!")
else:
    print(f"Angka {cari} tidak ditemukan.")
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int data[] = {10, 20, 30, 40, 50};
    int n = sizeof(data)/sizeof(data[0]);
    int cari;
    bool ditemukan = false;

    cout << "Masukkan angka yang dicari: ";
    cin >> cari;
```

```
    // Loop untuk mencari data
    for(int i = 0; i < n; i++){
        if(data[i] == cari){
            ditemukan = true;
            break;
        }
    }

    // Menampilkan hasil
    if(ditemukan)
        cout << "Angka " << cari << " ditemukan di
array!" << endl;
    else
        cout << "Angka " << cari << " tidak
ditemukan." << endl;

    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int[] data = {10, 20, 30, 40, 50};
        Scanner sc = new Scanner(System.in);
        boolean ditemukan = false;

        System.out.print("Masukkan angka yang dicari:
");
        int cari = sc.nextInt();

        // Loop untuk mencari data
        for(int i = 0; i < data.length; i++){
            if(data[i] == cari){
                ditemukan = true;
```

```
                break;
            }
        }

        // Menampilkan hasil
        if(ditemukan)
            System.out.println("Angka " + cari + "
ditemukan di array!");
        else
            System.out.println("Angka " + cari + "
tidak ditemukan.");
    }
}
```

PHP

```
<?php
$data = [10, 20, 30, 40, 50];
$cari = (int)readline("Masukkan angka yang dicari:
");
$ditemukan = false;

// Loop untuk mencari data
foreach($data as $d){
    if($d == $cari){
        $ditemukan = true;
        break;
    }
}

// Menampilkan hasil
if($ditemukan){
    echo "Angka $cari ditemukan di array!\n";
} else {
    echo "Angka $cari tidak ditemukan.\n";
}
?>
```

Penjelasan

- Looping digunakan untuk menelusuri seluruh elemen struktur data.
- Variabel boolean `ditemukan` digunakan untuk menandai apakah data yang dicari ada atau tidak.
- Break menghentikan perulangan lebih awal jika data sudah ditemukan, sehingga program lebih efisien.
- Struktur data berbeda di tiap bahasa, namun konsep pencarian sama: bandingkan setiap elemen dengan nilai yang dicari.

BAB 9
FUNGSI DAN PROSEDUR

A. Pendahuluan

Fungsi dan prosedur adalah blok kode yang dirancang untuk menjalankan tugas tertentu dalam program. Fungsi biasanya menghasilkan nilai kembali (*return value*), sedangkan prosedur lebih fokus pada eksekusi tugas tanpa mengembalikan nilai. Konsep ini penting karena memungkinkan program dibangun secara **modular**, sehingga setiap bagian kode dapat dipisahkan berdasarkan fungsinya masing-masing, membuat kode lebih mudah dibaca, dipelihara, dan dikembangkan, bahkan oleh programmer lain yang bekerja pada proyek yang sama.

Selain itu, penggunaan fungsi dan prosedur membantu mengurangi pengulangan kode (*code duplication*). Operasi yang sama dapat ditulis sekali sebagai fungsi dan dipanggil berkali-kali, sehingga mempermudah **debugging** dan pemeliharaan. Dengan merancang fungsi yang generik dan fleksibel, programmer juga dapat meningkatkan **reusability** dan membangun pustaka kode (*library*) yang dapat digunakan di berbagai program, meningkatkan efisiensi pengembangan perangkat lunak secara keseluruhan.

B. Konsep Modularisasi

Modularisasi adalah praktik dalam pemrograman untuk memecah program besar menjadi bagian-bagian yang lebih kecil dan terpisah, yang disebut modul atau fungsi (Rahmadi Asri et al., 2025). Setiap modul dirancang untuk menangani tugas atau fungsi tertentu secara spesifik, sehingga setiap bagian kode memiliki tanggung jawab yang jelas. Dengan

pendekatan ini, program tidak lagi ditulis sebagai satu blok besar yang sulit dipahami, melainkan menjadi struktur yang lebih teratur dan logis.

Keuntungan utama dari modularisasi adalah meningkatkan keterbacaan dan organisasi kode. Programmer dapat lebih mudah memahami alur program karena setiap modul memiliki tujuan yang jelas. Selain itu, modularisasi juga mempermudah proses debugging; ketika terjadi kesalahan, kita hanya perlu memeriksa modul terkait tanpa harus memeriksa seluruh program. Hal ini membuat identifikasi dan perbaikan bug menjadi lebih cepat dan efisien.

Selain itu, modularisasi mendukung penggunaan kembali kode (reusability). Modul yang telah dibuat dapat dipanggil kembali di bagian lain program atau bahkan di program lain tanpa menulis ulang kode dari awal. Hal ini tidak hanya menghemat waktu dan tenaga programmer, tetapi juga meningkatkan konsistensi dan keandalan kode. Dengan modularisasi, pengembangan program menjadi lebih efisien, terstruktur, dan mudah dipelihara, baik untuk proyek kecil maupun aplikasi berskala besar.

C. Fungsi dengan Parameter

Parameter memungkinkan kita mengirimkan informasi ke dalam fungsi, sehingga fungsi tersebut dapat memproses data yang berbeda-beda setiap kali dipanggil. Dengan parameter, satu fungsi dapat digunakan secara fleksibel untuk berbagai situasi tanpa harus menulis ulang kode. Hal ini tidak hanya meningkatkan efisiensi dan keterbacaan program, tetapi juga mendukung modularitas dan penggunaan kembali kode dalam berbagai konteks.

Contoh Fungsi Sederhana:

Python

```
def sapa(nama):
    print(f"Halo, {nama}!")

sapa("Andi")
sapa("Rina")
```

C++

```
#include <iostream>
using namespace std;

void sapa(string nama) {
    cout << "Halo, " << nama << "!" << endl;
}

int main() {
    sapa("Andi");
    sapa("Rina");
    return 0;
}
```

Java

```
public class Main {
    static void sapa(String nama) {
        System.out.println("Halo, " + nama + "!");
    }
    public static void main(String[] args) {
        sapa("Andi");
        sapa("Rina");
    }
}
```

PHP

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

sapa("Andi");
sapa("Rina");
?>
```

D. Argument Fungsi

Fungsi adalah blok kode yang dapat menerima argumen atau data dari pemanggilnya, sehingga fungsinya dapat memproses informasi yang berbeda-beda setiap kali dipanggil. Argumen memungkinkan fungsi menjadi lebih fleksibel, modular, dan dapat digunakan kembali. Secara umum, fungsi dapat menerima berbagai jenis argumen, yaitu positional, keyword (hanya di Python), default, dan variadic.

1. Positional Argument

Argumen positional adalah argumen yang diberikan sesuai urutan parameter saat mendefinisikan fungsi. Urutan ini harus diikuti agar nilai yang diberikan cocok dengan parameter yang dimaksud.

Contoh Python:

```
def sapa(nama, umur):
    print(f"Halo {nama}, umur kamu {umur} tahun.")

sapa("Andi", 20)
```

Contoh C++:

```
void sapa(string nama, int umur) {
    cout << "Halo " << nama << ", umur kamu " << umur
<< " tahun." << endl;
}
sapa("Andi", 20);
```

Contoh Java:

```
static void sapa(String nama, int umur) {
    System.out.println("Halo " + nama + ", umur kamu
" + umur + " tahun.");
}
sapa("Andi", 20);
```

```
    $y = 5; // lokal
    global $x;
    echo "Lokal: $y<br>";
    echo "Global: $x<br>";
}

tampil();
?>
```

F. Pernyataan Return

Fungsi dapat mengembalikan nilai ke pemanggil menggunakan return. Nilai yang dikembalikan ini kemudian dapat disimpan dalam variabel, digunakan dalam perhitungan, atau diproses lebih lanjut di bagian program lain. Pernyataan return membuat fungsi menjadi lebih fleksibel dan modular, karena bukan hanya menjalankan perintah di dalamnya, tetapi juga menyediakan hasil yang dapat dimanfaatkan kembali di luar fungsi. Dengan demikian, fungsi tidak hanya melakukan tugas, tetapi juga berperan sebagai blok kode yang menghasilkan informasi untuk program secara keseluruhan.

Python

```
def tambah(a, b):
    return a + b

hasil = tambah(10, 5)
print("Hasil:", hasil)
```

C++

```
#include <iostream>
using namespace std;

int tambah(int a, int b) {
    return a + b;
}
```

```
int main() {
    int hasil = tambah(10, 5);
    cout << "Hasil: " << hasil << endl;
    return 0;
}
```

Java

```
public class Main {
    static int tambah(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {
        int hasil = tambah(10, 5);
        System.out.println("Hasil: " + hasil);
    }
}
```

PHP

```
<?php
function tambah($a, $b){
    return $a + $b;
}
$hasil = tambah(10, 5);
echo "Hasil: $hasil";
?>
```

G. Prosedur (Fungsi Tanpa Return)

Prosedur adalah blok kode yang melakukan tugas tertentu tetapi tidak mengembalikan nilai ke pemanggil. Prosedur sering digunakan untuk mengeksekusi perintah seperti menampilkan output, memodifikasi variabel global, atau menjalankan operasi tanpa perlu menghasilkan nilai

balik. Meskipun tidak mengembalikan nilai, prosedur tetap bisa menerima parameter sehingga bisa fleksibel untuk berbagai situasi.

Contoh Prosedur: Menampilkan Pesan Selamat Datang

Python

```
def selamat_datang(nama):
    print(f"Selamat datang, {nama}!")

selamat_datang("Andi")
selamat_datang("Rina")
```

C++

```
#include <iostream>
using namespace std;

void selamatDatang(string nama) {
    cout << "Selamat datang, " << nama << "!" <<
endl;
}

int main() {
    selamatDatang("Andi");
    selamatDatang("Rina");
    return 0;
}
```

Java

```
public class Main {
    static void selamatDatang(String nama) {
        System.out.println("Selamat datang, " + nama
+ "!");
    }
    public static void main(String[] args) {
        selamatDatang("Andi");
        selamatDatang("Rina");
    }
}
```

PHP

```
<?php
function selamatDatang($nama){
    echo "Selamat datang, $nama!<br>";
}

selamatDatang("Andi");
selamatDatang("Rina");
?>
```

H. Docstring / Dokumentasi Fungsi

Dalam pemrograman, dokumentasi fungsi sangat penting untuk menjelaskan tujuan, cara kerja, dan parameter yang digunakan fungsi tersebut. Di Python, docstring adalah string yang ditulis di awal fungsi menggunakan tanda kutip tiga (`"""..."""`) dan berfungsi sebagai dokumentasi resmi fungsi. Docstring dapat diakses melalui atribut `__doc__` sehingga memudahkan programmer lain untuk memahami fungsi tanpa harus membaca seluruh kode.

Di bahasa lain, konsep serupa digunakan melalui komentar atau sistem dokumentasi khusus seperti JavaDoc di Java, yang memungkinkan deskripsi fungsi, parameter, dan nilai kembalian dicatat secara formal. Dokumentasi ini membantu programmer lain atau diri sendiri di masa depan untuk memahami fungsi, mempermudah pemeliharaan kode, serta meningkatkan keterbacaan dan profesionalitas program. Dengan adanya docstring atau dokumentasi yang baik, kode menjadi lebih jelas, terstruktur, dan mudah dipakai kembali.

Python

```
def tambah(a, b):
    """Fungsi ini menjumlahkan dua angka"""
    return a + b

help(tambah)
```

Java

```
/**
 * Fungsi ini menjumlahkan dua angka
 */
static int tambah(int a, int b) {
    return a + b;
}
```

I. Studi Kasus: Fungsi Kalkulator Sederhana

Program kalkulator sederhana menerima dua angka dan operator untuk menghasilkan hasil perhitungan.

Python

```
def kalkulator(a, b, operator):
    if operator == "+":
        return a + b
    elif operator == "-":
        return a - b
    elif operator == "*":
        return a * b
    elif operator == "/":
        return a / b
    else:
        return "Operator tidak valid"

print(kalkulator(10, 5, "+"))
print(kalkulator(10, 5, "/"))
```

C++

```
#include <iostream>
using namespace std;

double kalkulator(double a, double b, char op) {
```

```
    if(op == '+') return a + b;
    else if(op == '-') return a - b;
    else if(op == '*') return a * b;
    else if(op == '/') return a / b;
    else { cout << "Operator tidak valid"; return 0;
}
}

int main() {
    cout << kalkulator(10, 5, '+') << endl;
    cout << kalkulator(10, 5, '/') << endl;
    return 0;
}
```

Java

```
public class Main {
    static double kalkulator(double a, double b, char
op) {
        switch(op) {
            case '+': return a + b;
            case '-': return a - b;
            case '*': return a * b;
            case '/': return a / b;
            default: System.out.println("Operator
tidak valid"); return 0;
        }
    }
    public static void main(String[] args) {
        System.out.println(kalkulator(10, 5, '+'));
        System.out.println(kalkulator(10, 5, '/'));
    }
}
```

PHP

```
<?php
function kalkulator($a, $b, $op){
    switch($op){
        case '+': return $a + $b;
        case '-': return $a - $b;
        case '*': return $a * $b;
        case '/': return $a / $b;
        default: return "Operator tidak valid";
    }
}

echo kalkulator(10, 5, '+') . "<br>";
echo kalkulator(10, 5, '/') . "<br>";
?>
```

BAB 10
PEMROGRAMAN BERORIENTASI OBJEK (OOP)

A. Pendahuluan

Pemrograman Berorientasi Objek (Object-Oriented Programming / OOP) adalah pendekatan dalam membangun perangkat lunak dengan cara memodelkan komponen sistem sebagai objek yang saling berinteraksi. Setiap objek memiliki data dan perilaku, sehingga lebih mudah untuk menggambarkan fenomena dunia nyata secara langsung ke dalam program. OOP mempermudah pengembangan sistem skala besar karena kode menjadi lebih modular, lebih mudah dirawat, dan lebih fleksibel saat dikembangkan lebih lanjut.

B. Konsep Class dan Object

Pemrograman Berorientasi Objek (Object-Oriented Programming / OOP) menempatkan *class* dan *object* sebagai fondasi utama. Keduanya menjadi cara untuk merepresentasikan data dan perilaku suatu entitas dalam dunia nyata ke dalam bentuk program komputer. Dengan memahami perbedaan dan hubungan antara class dan object, programmer dapat membangun sistem yang lebih terstruktur, modular, dan mudah dikembangkan.

1. Class

Class adalah *cetak biru* (blueprint) atau *template* yang digunakan untuk merancang dan membangun objek dalam paradig ma Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP). Class mendefinisikan struktur dasar dan perilaku umum yang dimiliki oleh

d. PHP

```
<?php
class Mahasiswa {
    public $nama;
    public $nim;

    function tampilkanInfo() {
        echo "Nama: $this->nama, NIM: $this-
>nim";
    }
}

$mhs1 = new Mahasiswa();
$mhs1->nama = "Budi";
$mhs1->nim = "231045";
$mhs1->tampilkanInfo();
?>
```

Penjelasan :

1) `class Mahasiswa` mendefinisikan struktur data dan perilaku.
2) Pembuatan object dilakukan dengan:
   `$mhs1 = new Mahasiswa();`
3) Objek `$mhs1` sekarang memiliki atribut `nama` dan `nim`.
4) Pengisian atribut menggunakan operator ->:
   `$mhs1->nama = "Budi";`
5) Pemanggilan method juga memakai operator yang sama:
   `$mhs1->tampilkanInfo();`

C. Atribut & Method

Pemrograman Berorientasi Objek atau Object-Oriented Programming (OOP) merupakan pendekatan pemrograman yang berfokus pada pembuatan model perangkat lunak melalui objek-objek yang memiliki data dan perilaku. Setiap objek dibangun berdasarkan class, yaitu cetak biru (blueprint) yang mendefinisikan struktur serta kemampuan objek tersebut. Di dalam sebuah class, terdapat dua komponen fundamental, yaitu atribut dan method.

1. Atribut (Property / Field)

Atribut adalah elemen yang digunakan untuk menyimpan karakteristik atau keadaan internal dari sebuah objek. Atribut dapat berupa nilai sederhana, seperti angka dan teks, ataupun struktur kompleks seperti daftar, objek lain, atau tipe data buatan pengguna. Atribut inilah yang membuat setiap objek memiliki identitas unik meskipun berasal dari class yang sama.

Fungsi utama atribut:

- Menyimpan data yang merepresentasikan kondisi objek.
- Menggambarkan sifat atau properti yang membedakan satu objek dengan yang lain.
- Menjadi dasar bagi method untuk melakukan proses atau perhitungan.

Contoh pada class Mobil:

- **warna** → Menjelaskan penampilan kendaraan.
- **merk** → Mencerminkan identitas produsen.
- **kecepatan** → Menyatakan laju kendaraan pada saat tertentu.
- **jumlahRoda** → Menentukan jenis kendaraan berdasarkan konfigurasi rodanya.

Dengan atribut tersebut, setiap objek *Mobil* dapat memiliki karakteristik berbeda, misalnya mobil berwarna merah dengan merk Toyota dan kecepatan awal 0 km/jam.

2. **Method (Function / Behavior)**

   Method adalah elemen yang mendefinisikan apa saja yang dapat dilakukan oleh objek. Jika atribut berperan sebagai data, maka method berperan sebagai aksi atau logika yang memanipulasi data tersebut. Method memungkinkan objek untuk berinteraksi dengan dirinya sendiri, dengan objek lain, maupun dengan lingkungan program.

   Fungsi utama method:

   - Mengatur cara objek berperilaku.
   - Memungkinkan terjadinya perubahan pada atribut.
   - Mewakili tindakan yang menyerupai perilaku objek nyata.

   Contoh method pada class Mobil:

   - `mulai()` → Mengaktifkan mesin sehingga mobil siap digunakan.
   - `berhenti()` → Menghentikan laju mobil dan menurunkan kecepatan menjadi nol.
   - `percepat()` → Meningkatkan kecepatan mobil berdasarkan nilai tertentu.
   - `klakson()` → Menghasilkan bunyi sebagai bentuk interaksi dengan lingkungan.

   Method membuat objek tidak hanya menjadi kumpulan data, tetapi juga entitas yang aktif dan bisa melakukan sesuatu.

3. **Integrasi Atribut dan Method dalam OOP**

   Ketika atribut dan method digabungkan dalam sebuah class, terciptalah struktur yang menggambarkan objek dunia nyata secara lebih lengkap. Atribut menyediakan data yang mendeskripsikan "apa objek itu", sedangkan method mendefinisikan "apa yang bisa dilakukan objek tersebut." Pendekatan ini memungkinkan programmer membangun sistem perangkat lunak yang:

   - Lebih mudah dipahami karena mengikuti konsep nyata.
   - Lebih mudah dikembangkan dan diperluas.

- Mendorong modularitas karena setiap objek memiliki tanggung jawab spesifik.
- Meningkatkan keamanan melalui pengaturan akses data (enkapsulasi).

Misalnya, class *Mobil* dapat digunakan untuk membuat berbagai objek mobil dengan sifat dan perilaku masing-masing. Walaupun setiap objek berbeda, semuanya mengikuti pola yang sama sebagaimana ditentukan oleh class.

4. Contoh Atribut dan Method dalam OOP

a. Python

```
class Mobil:
    def __init__(self, merk, tahun):
        self.merk = merk       # atribut
        self.tahun = tahun

    def berjalan(self):        # method
        print(self.merk, "sedang berjalan")
```

Penjelasan

- Atribut (merk dan tahun) Dibuat di dalam konstruktor `__init__`.
- `self.merk` berarti atribut milik object, bukan variabel lokal.
- Method (**`berjalan()`**) Mengakses atribut object menggunakan `self`.
- Ketika dipanggil, method menggunakan data aktual dari object tersebut.
- Pembuatan object
  - `m1 = Mobil("Toyota", 2020)`
  - `m1.berjalan()`

  Output akan menyesuaikan merk object tersebut.

b. C++

```
class Mobil {
public:
    string merk;
    int tahun;

    void berjalan() {
        cout << merk << " sedang berjalan\n";
    }
};
```

Penjelasan

- Atribut (`merk` dan `tahun`) Dideklarasikan secara publik agar mudah diakses dari luar class (meski dalam praktik OOP modern, biasanya nanti akan dienkapsulasi).
- Method `(berjalan())` Mengakses atribut secara langsung karena berada dalam ruang lingkup class.
- Contoh penggunaan
  - `Mobil m1;`
  - `m1.merk = "Honda";`
  - `m1.tahun = 2018;`
  - `m1.berjalan();`

c. Java

```
class Mobil {
    String merk;
    int tahun;

    void berjalan() {
        System.out.println(merk + " sedang
berjalan");
    }
}
```

Penjelasan

- Atribut, biasanya dibiarkan akses default atau dibuat private (nanti dibahas pada encapsulation).
- Method `(berjalan())` Mengakses atribut langsung.
- Contoh penggunaan
  - `Mobil m1 = new Mobil();`
  - `m1.merk = "Yamaha";`
  - `m1.tahun = 2021;`
  - `m1.berjalan();`

d. PHP

```
class Mobil {
    public $merk;
    public $tahun;

    function berjalan() {
        echo "$this->merk sedang berjalan\n";
    }
}
```

Penjelasan

- Atribut, Dituliskan dengan awalan `$`, dan `$this->` digunakan untuk mengaksesnya dari dalam class.
- Method `(berjalan())`, Menggunakan `$this->merk` untuk mengambil nilai atribut object.
- Contoh penggunaan
  - `$m1 = new Mobil();`
  - `$m1->merk = "Suzuki";`
  - `$m1->tahun = 2019;`
  - `$m1->berjalan();`

5. Apa yang Bisa Dipahami dari Keempat Bahasa Ini?

Meskipun sintaks berbeda, konsepnya sama:

a. Atribut
   - Menyimpan data object
   - Bersifat spesifik untuk tiap object
   - Diakses dengan:
     - `self` di Python
     - akses langsung atau `this` di C++
     - `this` di Java
     - `$this` di PHP

b. Method
   - Fungsi yang "menempel" pada object
   - Dapat membaca atau mengubah atribut
   - Dipanggil melalui object, bukan class (kecuali method statis yang juga akan dibahas nanti pada bab lanjutan)
   - Contoh pemanggilan method sama pola di semua bahasa:
     - `object.method()`
     - `object->method()  // PHP`

6. Transisi ke Subbab Berikutnya

Sampai di sini, kita sudah memahami bahwa atribut dan method bisa diakses secara langsung. Namun, praktik OOP modern tidak selalu memperbolehkan akses langsung terhadap atribut, karena dapat menyebabkan data tidak konsisten atau mudah berubah tanpa kontrol. Untuk mengatasi hal tersebut, OOP menggunakan konsep Encapsulation, yaitu penyembunyian data dan pemberian akses yang terkontrol.

D. Encapsulation (Enkapsulasi)

1. Devinisi Encapsulation (Enkapsulasi)

Encapsulation adalah konsep dalam OOP yang menyatukan data (atribut) dan fungsi yang bekerja pada data tersebut (method) dalam satu kesatuan — yaitu class — serta membatasi akses langsung ke data tersebut atau melindungi data agar tidak dapat diakses atau diubah secara sembarangan dari luar class. Tujuannya adalah melindungi data agar tidak diubah secara sembarangan, hanya bisa melalui method khusus (getter & setter).

Dengan enkapsulasi:

- Atribut dilindungi menggunakan visibility seperti private atau protected.
- Akses ke atribut dilakukan melalui method khusus (getter dan setter).
- Programmer dapat mengontrol cara data diubah, sehingga data tetap valid dan konsisten.

Tujuan Enkapsulasi

- Melindungi data dari perubahan yang tidak sah.
- Mencegah manipulasi langsung yang dapat merusak struktur object.
- Memberikan interface yang jelas kepada pengguna class.
- Mempermudah pemeliharaan kode.
- Mendukung modularitas dan penyembunyian detail implementasi.

Analogi Kehidupan Nyata

Enkapsulasi sama seperti:

- ATM
  Pengguna tidak boleh mengakses komponen mekanik di dalamnya; hanya boleh memakai tombol dan kartu untuk “mengontrol” proses.
- Smartphone
  Banyak fungsi internal tersembunyi. Pengguna hanya berinteraksi melalui antarmuka (UI), bukan langsung ke rangkaian dalam.

2. Prinsip Enkapsulasi

a. Membatasi Akses (Access Modifier / Visibility)

Tabel 10. 2 Access Modifier Public, Protected and Private

Bahasa	Public	Protected	Private
Python	Tidak ketat (konvensi _ dan __)	Tidak ketat	Private pseudo-name-mangling
C++	✔	✔	✔
Java	✔	✔	✔
PHP	✔	✔	✔

Level akses ini menentukan apakah atribut/method dapat diakses:
Public → bisa diakses dari mana saja.
Protected → hanya diakses oleh class dan turunannya.
Private → hanya diakses oleh class itu sendiri.

b. Data Hiding (Penyembunyian Data)
Atribut dibuat private sehingga tidak bisa diakses langsung dari luar.

c. Controlled Access (Akses Terkendali)
Akses ke atribut dilakukan melalui method khusus:
- Getter: mengambil nilai atribut
- Setter: mengubah nilai atribut dengan kontrol tertentu

d. Keamanan Data
Mencegah hal-hal seperti:
- nilai atribut salah atau tidak valid,
- perubahan nilai tanpa aturan.

3. Contoh Inheritance dalam 4 Bahasa Pemrograman

a. Python

```
# Parent Class
class Hewan:
    def suara(self):
        return "Hewan bersuara"

# Child Class
class Kucing(Hewan):
    def suara(self):
        return "Meong"

# Pemanggilan
obj = Kucing()
print(obj.suara())
```

b. C++

```
#include <iostream>
using namespace std;

class Hewan { // Parent
public:
    virtual string suara() {
        return "Hewan bersuara";
    }
};

class Kucing : public Hewan { // Child
public:
    string suara() override {
        return "Meong";
    }
};

int main() {
    Kucing k;
    cout << k.suara();
}
```

c. Java

```
class Hewan { // Parent
    String suara() {
        return "Hewan bersuara";
    }
}

class Kucing extends Hewan { // Child
    @Override
    String suara() {
        return "Meong";
    }
}

public class Main {
    public static void main(String[] args) {
        Kucing k = new Kucing();
        System.out.println(k.suara());
    }
}
```

d. PHP

```
<?php
class Hewan { // Parent
    public function suara() {
        return "Hewan bersuara";
    }
}

class Kucing extends Hewan { // Child
    public function suara() {
        return "Meong";
    }
}

$k = new Kucing();
echo $k->suara();
?>
```

4. Jenis-Jenis Inheritance

Tabel 10. 5 Jenis-jenis Inheritance

Jenis	Deskripsi	Contoh
Single Inheritance	Satu child mewarisi satu parent	Kucing → Hewan
Multilevel Inheritance	Child menjadi parent untuk child lain	Hewan → Mamalia → Kucing
Hierarchical Inheritance	Satu parent memiliki banyak child	Hewan → Kucing, Hewan → Anjing
Multiple Inheritance	Satu child mewarisi banyak parent	Hanya ada di Python & C++ (Java & PHP tidak mendukung langsung)

5. Overriding pada Inheritance

Overriding adalah proses menulis ulang method parent di dalam child untuk memberikan perilaku yang berbeda.

Contoh singkat Python:

```
class A:
    def pesan(self):
        return "Pesan dari A"

class B(A):
    def pesan(self):
        return "Pesan dari B"  # Overriding
```

6. Kapan Inheritance Harus Digunakan?

Gunakan inheritance ketika:

✔ Ada hubungan **is-a** (Kucing *is a* Hewan)

✔ Child memang memperluas perilaku parent

✔ Banyak child memiliki fungsi dasar yang sama

Jangan gunakan inheritance ketika:

✗ Hubungannya bukan “is-a”, tapi “has-a” → gunakan **komposisi**

✗ Hanya untuk berbagi fungsi tanpa hierarki jelas

F. Polymorphism (Polimorfisme)

1. Pengertian Polymorphism

Polymorphism adalah kemampuan suatu objek untuk memiliki banyak bentuk (poly = banyak, morph = bentuk).

Dalam OOP, polymorphism memungkinkan:

- Method yang sama dapat memiliki perilaku berbeda
- Bergantung pada objek mana yang memanggilnya (parent atau child)

Dengan kata lain, polymorphism membuat program menjadi fleksibel, mudah dikembangkan, dan lebih sederhana dalam penulisan.

2. Kenapa Polymorphism Penting?

Tabel 10. 6 Manfaat Inheritance

Manfaat	Penjelasan
Fleksibilitas kode	Satu interface bisa digunakan untuk berbagai implementasi.
Mempermudah ekspansi	Saat menambah class baru, tidak perlu mengubah kode lama.
Memperjelas struktur OOP	Kode lebih rapi dan modular.
Dukungan konsep overriding	Child dapat membuat implementasi berbeda dari parent.

3. Jenis-Jenis Polymorphism

a. Compile-Time Polymorphism (Static Binding)

Terjadi saat compile.

Contoh: Method Overloading (nama method sama, parameter berbeda).

➔ Tersedia di Java, C++, tidak di Python dan sebagian di PHP.

b. Run-Time Polymorphism (Dynamic Binding)

Terjadi saat program berjalan.

Contoh: Method Overriding (child menimpa method parent).

➔ Ada di semua bahasa OOP modern: Python, Java, C++, PHP.

4. Contoh Polymorphism dalam 4 Bahasa

a. Python (Run-time Polymorphism)

```
class Hewan:
    def suara(self):
        return "Suara hewan"

class Kucing(Hewan):
    def suara(self):
        return "Meong"

class Anjing(Hewan):
    def suara(self):
        return "Guk"

def cetak_suara(hewan):
    print(hewan.suara())

cetak_suara(Kucing())
cetak_suara(Anjing())
```

b. C++ (Compile-time & Run-time)

1) Overloading (Compile-time)

```
#include <iostream>
using namespace std;

class Matematika {
public:
    int tambah(int a, int b) {
        return a + b;
    }

    double tambah(double a, double b) {
        return a + b;
    }
};
```

b. C++ (Abstract Class menggunakan pure virtual function)

```
#include <iostream>
using namespace std;

class Kendaraan {
public:
    virtual void bergerak() = 0;  // pure
virtual
};

class Mobil : public Kendaraan {
public:
    void bergerak() {
        cout << "Mobil bergerak di jalan raya"
<< endl;
    }
};

int main() {
    Kendaraan* k = new Mobil();
    k->bergerak();
}
```

- ✔ = 0 menandai method abstrak
- ✔ Class dengan method seperti ini menjadi abstrak

c. Java (Abstract Class)

```
abstract class Kendaraan {
    abstract void bergerak();
}

class Mobil extends Kendaraan {
    void bergerak() {
        System.out.println("Mobil bergerak di
jalan raya");
    }
}

public class Main {
    public static void main(String[] args) {
        Kendaraan k = new Mobil();
```

```
        k.bergerak();
    }
}
```

- ✔ Class abstrak tidak dapat langsung dibuat objek
- ✔ Subclass wajib override method abstrak

d. PHP (Abstract Class)

```
<?php
abstract class Kendaraan {
    abstract public function bergerak();
}

class Mobil extends Kendaraan {
    public function bergerak() {
        echo "Mobil bergerak di jalan raya";
    }
}

$k = new Mobil();
$k->bergerak();
?>
```

- ✔ PHP menggunakan `abstract` untuk class dan method
- ✔ Subclass harus mengimplementasikan method abstrak

H. Studi Kasus OOP: Sistem Manajemen Mahasiswa

Dengan Class Diagram + Implementasi Python, C++, Java, dan PHP

1. Deskripsi Studi Kasus

Dalam studi kasus ini, kita membangun Sistem Manajemen Mahasiswa sederhana berbasis OOP. Sistem ini harus dapat:

a. Menyimpan data mahasiswa

- Nama
- NIM

- Jurusan
- IPK

b. Memiliki perilaku (method)

- menampilkan informasi mahasiswa
- menghitung status kelulusan (lulus/tidak)

c. Mendemonstrasikan konsep OOP

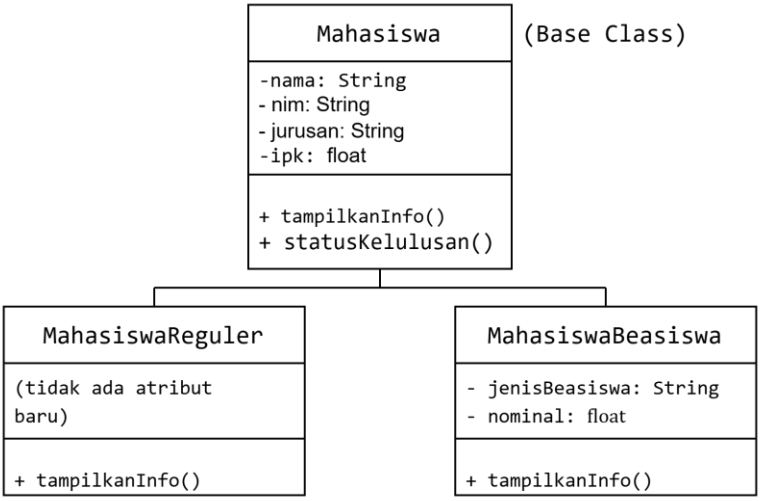
- class & object
- atribut & method
- inheritance
- polymorphism
- encapsulation
- abstraction (opsional)

d. Mendukung jenis mahasiswa berbeda, misalnya:

- Mahasiswa Reguler
- Mahasiswa Beasiswa

  (ada atribut tambahan: jumlah beasiswa / syarat IPK)

2. Class Diagram



Gambar 10. 2 Class Diagram

3. Implementasi di Berbagai Bahasa

a. Python Implementation

```
class Mahasiswa:
    def __init__(self, nama, nim, jurusan,
ipk):
        self.nama = nama
        self.nim = nim
        self.jurusan = jurusan
        self.ipk = ipk

    def tampilkan_info(self):
        print(f"Nama: {self.nama}, NIM:
{self.nim}")
        print(f"Jurusan: {self.jurusan}, IPK:
{self.ipk}")

    def status_kelulusan(self):
        return "Lulus" if self.ipk >= 2.75 else
"Tidak Lulus"


class MahasiswaReguler(Mahasiswa):
    pass


class MahasiswaBeasiswa(Mahasiswa):
    def __init__(self, nama, nim, jurusan, ipk,
jenis, nominal):
        super().__init__(nama, nim, jurusan,
ipk)
        self.jenis = jenis
        self.nominal = nominal

    def tampilkan_info(self):
        super().tampilkan_info()
        print(f"Beasiswa: {self.jenis},
Nominal: {self.nominal}")


# Penggunaan
m1 = MahasiswaReguler("Budi", "231045",
"Informatika", 3.10)
```

```
m2 = MahasiswaBeasiswa("Siti", "231122",
"Sistem Informasi", 3.80, "Prestasi", 1500000)

m1.tampilkan_info()
print(m1.status_kelulusan())

print("---")

m2.tampilkan_info()
print(m2.status_kelulusan())
```

b. C++ Implementation

```
#include <iostream>
using namespace std;

class Mahasiswa {
public:
    string nama, nim, jurusan;
    float ipk;

    Mahasiswa(string n, string ni, string j,
float i)
        : nama(n), nim(ni), jurusan(j), ipk(i)
{}

    virtual void tampilkanInfo() {
        cout << "Nama: " << nama << ", NIM: "
<< nim << endl;
        cout << "Jurusan: " << jurusan << ",
IPK: " << ipk << endl;
    }

    string statusKelulusan() {
        return (ipk >= 2.75) ? "Lulus" : "Tidak
Lulus";
    }
};

class MahasiswaReguler : public Mahasiswa {
public:
    MahasiswaReguler(string n, string ni,
string j, float i)
```

Dengan memahami cara menangani error dan melakukan debugging secara efektif, programmer dapat:

- Menghasilkan program yang lebih stabil dan mudah dirawat.
- Menjadi lebih cepat dalam mengidentifikasi masalah.
- Meningkatkan kualitas kode dan mengurangi potensi bug.

F. Proses Debugging yang Sistematis

Debugging bukan sekadar mencari error, tetapi proses analisis yang terstruktur:

Tahapan debugging profesional:

1. Mengidentifikasi masalah
   - Membaca pesan error
   - Melihat gejala program
2. Mengumpulkan informasi
   - Input pengguna
   - Nilai variabel saat error
   - Log sistem
3. Mengisolasi kode
   - Menonaktifkan bagian yang tidak terkait
   - Memeriksa fungsi satu per satu
4. Menganalisis kemungkinan penyebab
   - Tipe data?
   - Alur logika?
   - Dependency?
5. Menerapkan perbaikan
   - Memperbaiki baris bermasalah
   - Memodifikasi alur logika
6. Menguji ulang (regression testing)
   - Menjalankan kembali program setelah perbaikan
   - Memastikan error baru tidak muncul

Ilustrasi Alur Debugging:

```
Error → Identifikasi → Isolasi → Perbaikan → Pengujian → Selesai
```

G. Teknik Debugging Sederhana

1. Menampilkan nilai variabel (print debugging)
   Python
   ```
   print("nilai x =", x)
   ```
   C++
   ```
   cout << "nilai x = " << x << endl;
   ```
2. Komentar sebagian kode
   Mematikan sebagian blok untuk mengetahui letak masalah.
3. Menggunakan breakpoint
   Memberhentikan eksekusi pada titik tertentu.
   Dukungan IDE seperti VSCode, PyCharm, Eclipse.
4. Trace alur eksekusi (manual atau tools)
   Menelusuri alur eksekusi program secara manual atau otomatis.

H. Tools Debugging pada Berbagai Bahasa

Berbagai bahasa pemrograman memiliki alat resmi:
Python

- `pdb`
- Modul `logging`

Contoh logging python
```
import logging
logging.basicConfig(level=logging.DEBUG)
logging.debug("Nilai x adalah %d", x)
```

Java

- `jdb`
- `Debugger` bawaan IntelliJ, Eclipse, NetBeans

C++

- `gdb`
- `valgrind` (deteksi memory leak)

PHP

- Xdebug (breakpoint, tracing, profiling)

Digunakan untuk tracing, profiling, breakpoint.

I. Debugging pada Program Berorientasi Objek

OOP memiliki tantangan debugging yang berbeda:

1. Error karena inheritance

```
class A {
    void tampil() { System.out.println("A"); }
}

class B extends A {
    int x;  // belum diinisialisasi
}
```

2. Method overriding sala

```
class Hewan:
    def suara(self):
        print("suara umum")

class Kucing(Hewan):
    def suara(self):  # benar
        print("meong")

# salah: nama method beda
def suaraa(self):
    print("meong")
```

3. Null reference / objek belum dibuat

```
Mobil m;
m.berjalan();   // NullPointerException
```

4. Kesalahan akses atribut (encapsulation)

```
class Data {
    private $nilai = 10;
}

$obj = new Data();
echo $obj->nilai; // error, tidak bisa akses
private
```

J. Logging dan Monitoring

Logging membantu memantau perilaku program secara otomatis.
Level Log Umum:

- DEBUG → informasi detail, detail teknis
- INFO → status umum, status normal
- WARNING → potensi masalah
- ERROR → kesalahan yang dapat ditangani
- CRITICAL → kesalahan fatal

Python

```
logging.error("File tidak ditemukan!")
```

PHP

```
error_log("Terjadi error pada koneksi database");
```

K. Exception Handling & Custom Exception

Exception handling memastikan program tetap berjalan meski terjadi error.

Tujuan exception handling

- Menangani error tanpa menghentikan program
- Memberikan pesan error yang lebih jelas

Python

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Tidak boleh bagi nol")
```

Custom Exception

```
class NilaiError(Exception):
    pass
```

Java

```
try {
    int x = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Error: " + e.getMessage());
}
```

Custom exception

```
class DataException extends Exception {
    public DataException(String msg) {
        super(msg);
    }
}
```

C++

```
try {
    throw "Error terjadi";
} catch(const char* msg) {
    cout << msg;
}
```

dimiliki. Selain itu, programmer juga perlu memperhatikan keamanan data, performa, serta penanganan error agar interaksi dengan file berjalan efisien dan bebas dari gangguan.

2. Peranan File Handling dalam Pengembangan Software

File handling memiliki peranan krusial dalam berbagai konteks, antara lain:

a. Penyimpanan Data Permanen
   File berfungsi sebagai tempat menyimpan informasi yang harus tetap tersedia meskipun program telah berhenti berjalan. Dengan adanya penyimpanan permanen, data penting seperti laporan, catatan transaksi, atau hasil proses komputasi dapat diakses kembali tanpa perlu menjalankan ulang perhitungan atau input data.
b. Komunikasi Antar Sistem
   File sering dijadikan medium pertukaran data antar aplikasi maupun antar server. Format standar seperti CSV, JSON, dan XML memudahkan sistem berbeda—yang mungkin menggunakan bahasa atau platform berbeda—untuk memahami data yang sama karena struktur dan aturannya telah disepakati secara luas.
c. Pengolahan Media
   Dalam aplikasi multimedia, file biner menyimpan informasi kompleks seperti pixel gambar, sampel audio, atau frame video. Program grafis, editor musik, maupun perangkat lunak pengolah video memanfaatkan file-file tersebut untuk menampilkan, memodifikasi, atau mengekspor hasil karya dalam berbagai format.
d. Manajemen Konfigurasi
   Banyak program mengandalkan file konfigurasi untuk menentukan perilaku atau pengaturan awal saat dijalankan. File seperti .ini, .json, atau .yaml digunakan untuk menyimpan parameter, preferensi pengguna, hingga kredensial koneksi sehingga aplikasi dapat beroperasi sesuai kebutuhan tanpa perlu dikonfigurasi ulang setiap kali.

e. Pemrosesan Data Berskala Besar
   Pada bidang data science dan machine learning, file digunakan untuk menyimpan dataset berukuran besar yang berisi ribuan hingga jutaan entri. Data ini kemudian diolah, dianalisis, atau dilatih menggunakan model tertentu, sehingga file menjadi fondasi penting dalam alur kerja komputasi berkapasitas tinggi.

3. Struktur Umum File dalam Sistem Operasi

Struktur umum file dalam sistem operasi mencakup beberapa elemen yang memungkinkan komputer mengenali, mengelola, dan menampilkan informasi yang tersimpan. Dalam sistem operasi, file terdiri dari:

a. Nama file (filename)
b. Ekstensi file
c. Metadata: waktu modifikasi, ukuran, hak akses
d. Isi file: data aktual

Setiap file diawali dengan nama file (filename) yang berfungsi sebagai identitas utama di dalam direktori. Nama ini biasanya diikuti oleh ekstensi file, yaitu penanda jenis atau format berkas, misalnya .txt, .jpg, atau .exe, sehingga sistem dan pengguna dapat mengetahui jenis data yang terkandung di dalamnya.

Selain penamaan, file juga menyimpan metadata, yaitu informasi deskriptif yang tidak termasuk dalam konten utama tetapi sangat penting untuk pengelolaan file. Metadata meliputi waktu pembuatan dan modifikasi, ukuran file, tipe file, serta pengaturan hak akses yang menentukan siapa saja yang boleh membaca, menulis, atau mengeksekusi file tersebut. Metadata ini dimanfaatkan sistem operasi untuk melakukan proses seperti penjadwalan, indexing, hingga pengaturan keamanan.

Di luar bagian-bagian tersebut, file juga memiliki isi file, yaitu data aktual yang ingin disimpan pengguna, baik berupa teks, gambar, audio, program, maupun bentuk data lainnya. Ketika file diakses oleh program, sistem operasi menggunakan file descriptor atau file handle—sebuah identitas unik yang diberikan kepada setiap file yang sedang dibuka. Struktur ini memungkinkan sistem operasi melacak operasi apa saja yang

boleh dilakukan terhadap file dan memastikan setiap proses yang mengaksesnya bekerja secara aman dan teratur.

B. Jenis-Jenis File

File dalam ilmu komputer dikategorikan berdasarkan representasi internalnya.

1. File Teks (Text File)

Karakteristik:

- Disimpan sebagai rangkaian karakter (ASCII/UTF-8).
- Dapat dibaca manusia menggunakan text editor.
- Digunakan untuk catatan, konfigurasi, log, hingga struktur data sederhana seperti CSV.

Contoh Format Text File:

- .txt – dokumen teks standar
- .csv – data tabel
- .json – format objek ringan
- .html – dokumen web
- .xml – data terstruktur

Kelebihan:

- Mudah dibaca dan ditulis
- Portable pada berbagai perangkat
- Cocok untuk debugging

Kekurangan:

- Tidak mendukung struktur kompleks
- Tidak efisien untuk data besar

2. File Biner (Binary File)

Karakteristik:

- Disimpan dalam bentuk byte mentah.
- Tidak dapat dibaca langsung oleh manusia.
- Umumnya digunakan untuk media dan program.

Contoh Format File Biner:

- .jpg, .png – gambar
- .mp3, .wav – audio
- .mp4, .avi – video
- .exe, .dll – executable
- .dat, .bin – data biner generik

Kelebihan:

- Lebih efisien
- Dapat menyimpan struktur kompleks
- Akses file lebih cepat

Kekurangan:

- Sulit dianalisis manual
- Rentan rusak jika salah membaca/menulis

C. Operasi Dasar dalam File Handling

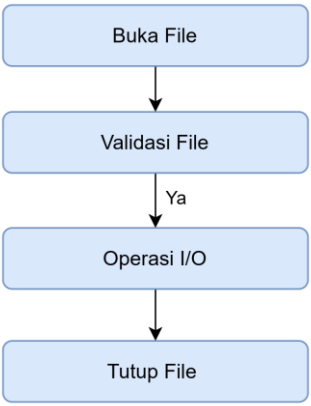
1. Tahapan Umum Operasi File

Tahapan umum operasi file adalah langkah-langkah dasar yang harus dilakukan sebuah program ketika ingin berinteraksi dengan file di penyimpanan permanen. Setiap bahasa pemrograman umumnya mengikuti urutan yang sama: file harus dibuka terlebih dahulu, kemudian program melakukan proses input/output seperti membaca atau menulis data, dan akhirnya file ditutup kembali agar perubahan tersimpan dengan aman serta sumber daya sistem dapat dilepas. Tahapan ini memastikan pengolahan file berjalan terstruktur, efisien, dan terhindar dari kesalahan.

Setiap interaksi dengan file umumnya mengikuti tiga tahap utama yang menjadi alur dasar proses pengelolaan file:

a. Membuka file
b. Melakukan operasi I/O (Input/Output)
c. Menutup file

Diagram umum operasi file adalah sebagai berikut:



Gambar 12. 1 Diagram umum operasi file

Flowchart tersebut menggambarkan urutan prosedural yang umum digunakan dalam pemrosesan file pada sebuah program. Tahap pertama adalah Buka File, yaitu proses ketika program mencoba mengakses atau membuka file yang diperlukan. Langkah ini merupakan pintu awal yang menentukan apakah proses berikutnya dapat dilanjutkan. Jika file tidak dapat dibuka, biasanya program akan berhenti atau masuk ke mekanisme penanganan kesalahan, tetapi dalam diagram ini fokusnya adalah alur ideal.

Tahap berikutnya adalah Validasi File, yaitu proses pemeriksaan terhadap file yang telah dibuka. Pada langkah ini, program memastikan bahwa file benar-benar tersedia, formatnya benar, tidak kosong, dan memiliki izin akses yang sesuai. Validasi sangat penting karena menentukan apakah data dalam file layak diproses lebih lanjut. Jika file tidak valid, program seharusnya melakukan error handling, namun dalam diagram ini ditekankan alur utama tanpa percabangan.

Setelah file divalidasi, program melanjutkan ke tahap Operasi I/O, yaitu proses membaca atau menulis data pada file sesuai dengan kebutuhan aplikasi. Operasi ini merupakan inti dari manipulasi file. Setelah selesai melakukan operasi I/O, program memasuki tahap terakhir yaitu Tutup File, yang bertujuan melepaskan sumber daya sistem dan

memastikan file tidak terkunci atau menyebabkan kebocoran memori. Menutup file merupakan bagian penting dari praktik pemrograman yang aman dan terstruktur.

2. Mode Pembukaan File dalam Empat Bahasa

Tabel 12. 1 Mode Pembukaan File dalam Empat Bahasa

Mode	Fungsi	Python	Java	C++	PHP
Read	Membaca file	`"r"`	FileReader	`ifstream`	`"r"`
Write	Menulis & menimpa file	`"w"`	FileWriter	`ofstream`	`"w"`
Append	Menambah isi file	`"a"`	FileWriter(true)	`ios::app`	`"a"`
Read+ Write	Baca dan tulis	`"r+"`	RandomAccessFile	`fstream`	`"r+"`
Binary Read	Membaca file biner	`"rb"`	InputStream	`ios::binary`	`"rb"`
Binary Write	Menulis file biner	`"wb"`	OutputStream	`ios::binary`	`"wb"`

3. Contoh Pembukaan File dalam Empat Bahasa

Python

```
f = open("data.txt", "r")
f.close()
```

Java

```
FileReader fr = new FileReader("data.txt");
fr.close();
```

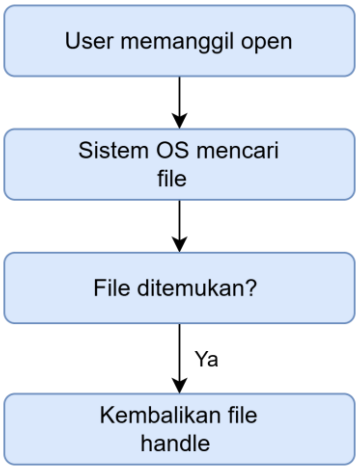
C++

```
ifstream file("data.txt");
file.close();
```

PHP

```
$file = fopen("data.txt", "r");
fclose($file);
```

4. Diagram Alur Pembukaan File (Dengan Validasi)



Gambar 12. 2 Diagram alur pembukaan file

Flowchart tersebut menggambarkan alur lengkap proses pembacaan file dalam sebuah program. Proses dimulai dari langkah Buka File, yaitu ketika program mencoba mengakses file yang dibutuhkan. Pada tahap ini, program harus memulai koneksi ke file dan menyiapkan seluruh mekanisme pembacaan data. Namun, sebelum dapat melanjutkan, program perlu memastikan apakah file benar-benar dapat diakses atau tidak, sehingga dibutuhkan proses pengecekan kondisi.

Setelah file dibuka, program masuk ke langkah File Tersedia? yang merupakan percabangan (decision). Pada titik ini, program mengevaluasi apakah file yang dimaksud ada, dapat dibaca, dan tidak mengalami masalah seperti rusak atau tidak memiliki izin akses. Jika hasil pengecekan menunjukkan bahwa file tidak tersedia, alur berpindah ke proses Error Handling, yaitu penanganan kesalahan untuk mencegah program crash. Namun, jika file tersedia dan dapat dibuka dengan benar, program melanjutkan ke proses berikutnya, jika Tidak, maka sistem mengembalikan exception / warning.

F. File Teks vs File Biner

File teks (text file) dan file biner (binary file) merupakan dua format penyimpanan data yang umum digunakan dalam pemrograman, masing-masing memiliki karakteristik, keunggulan, dan konteks penggunaan yang berbeda. File teks digunakan untuk data yang bisa dibaca manusia, sedangkan file biner digunakan untuk media atau struktur kompleks. Dalam file teks, data disimpan dalam bentuk karakter sehingga isinya dapat dilihat dan dipahami langsung melalui editor teks biasa. Format ini umumnya menggunakan encoding seperti ASCII atau UTF-8 dan banyak dimanfaatkan untuk keperluan konfigurasi, dokumentasi, atau penyimpanan data sederhana.

Di sisi lain, file biner menyimpan data dalam bentuk byte mentah yang dirancang untuk dibaca oleh program, bukan manusia. Struktur biner memungkinkan representasi informasi yang lebih efisien dan mendukung penyimpanan berbagai jenis data yang lebih kaya, seperti gambar, audio, video, atau objek data kompleks dalam aplikasi. Karena sifatnya yang tidak terbaca secara langsung, file biner memerlukan prosedur dan fungsi khusus untuk memprosesnya dengan benar.

Perbedaan mendasar antara kedua jenis file ini terletak pada cara penyimpanan dan akses datanya. File teks menawarkan kemudahan pembacaan dan debugging, namun kurang efisien untuk data yang berukuran besar maupun yang memerlukan format ketat. Sebaliknya, file biner memberikan kinerja dan fleksibilitas lebih baik, meskipun memerlukan kontrol yang lebih cermat dalam pengolahan.

Selain itu, pemilihan antara file teks dan file biner juga bergantung pada tujuan aplikasi. Untuk aplikasi yang membutuhkan interoperabilitas tinggi, seperti pertukaran data antarsistem atau penyimpanan konfigurasi, file teks sering menjadi pilihan karena mudah dibaca dan dimodifikasi. Namun untuk aplikasi yang mengelola media, komputasi numerik, atau struktur data kompleks, file biner jauh lebih sesuai karena dapat mempertahankan format data asli dan meminimalkan ukuran penyimpanan. Dengan mempertimbangkan kebutuhan tersebut,

programmer dapat menentukan jenis file yang paling efektif untuk digunakan.

Contoh Membaca File Biner

1. Python

```
with open("gambar.jpg", "rb") as f:
    data = f.read()
```

2. C++

```
ifstream file("gambar.jpg", ios::binary);
```

3. Java

```
FileInputStream fis = new
FileInputStream("gambar.jpg");
byte[] buffer = new byte[1024];
int read;
while ((read = fis.read(buffer)) != -1) {
    // proses data
}
fis.close();
```

4. PHP

```
<?php
$data = file_get_contents("gambar.jpg");
?>
```

G. Manajemen File dan Direktori

Manajemen file dan direktori merupakan serangkaian operasi yang digunakan untuk mengelola struktur berkas dalam sistem komputer. Operasi ini mencakup pengecekan keberadaan file, penghapusan, penyalinan, pemindahan, penggantian nama, hingga pembuatan folder baru. Kemampuan melakukan manajemen file sangat penting dalam pemrograman karena banyak aplikasi memerlukan pengaturan data yang tersimpan di penyimpanan permanen.

Secara umum, operasi manajemen file bertujuan memastikan data dapat tertata dengan baik, mudah diakses, serta dapat dimodifikasi sesuai kebutuhan program. Berbagai bahasa pemrograman menyediakan fungsi atau library khusus untuk mempermudah pengelolaan file dan direktori. Berikut contoh operasi dasar dalam beberapa bahasa populer.

1. Python (menggunakan modul `os` dan `shutil`)

```
import os
import shutil

# Cek apakah file ada
print(os.path.exists("data.txt"))

# Hapus file
os.remove("data.txt")

# Salin file
shutil.copy("sumber.txt", "salinan.txt")

# Pindah file
shutil.move("salinan.txt", "folder_baru/salinan.txt")

# Rename file
os.rename("lama.txt", "baru.txt")

# Buat folder baru
os.mkdir("folder_baru")
```

2. Java (menggunakan kelas `File` dan `Files`)

```
import java.io.File;
import java.nio.file.*;

public class FileOps {
    public static void main(String[] args) throws
Exception {
        File file = new File("data.txt");

        // Cek file
        System.out.println(file.exists());
```

```
        // Hapus
        file.delete();

        // Salin
        Files.copy(Paths.get("sumber.txt"),
Paths.get("salinan.txt"));

        // Pindah
        Files.move(Paths.get("salinan.txt"),
Paths.get("folder/salinan.txt"));

        // Rename
        file.renameTo(new File("baru.txt"));

        // Buat folder
        new File("folder").mkdir();
    }
}
```

3. C++ (menggunakan `<filesystem>`)

```
#include <iostream>
#include <filesystem>
using namespace std;
namespace fs = std::filesystem;

int main() {
    // Cek file
    cout << fs::exists("data.txt") << endl;

    // Hapus file
    fs::remove("data.txt");

    // Salin file
    fs::copy("sumber.txt", "salinan.txt");

    // Pindah file
    fs::rename("salinan.txt", "folder/salinan.txt");

    // Rename file
    fs::rename("lama.txt", "baru.txt");

    // Buat folder
```

```
    fs::create_directory("folder");

    return 0;
}
```

4. PHP (menggunakan fungsi standar file system)

```
<?php
// Cek file
echo file_exists("data.txt");

// Hapus file
unlink("data.txt");

// Salin file
copy("sumber.txt", "salinan.txt");

// Pindah file
rename("salinan.txt", "folder/salinan.txt");

// Rename file
rename("lama.txt", "baru.txt");

// Buat folder
mkdir("folder");
?>
```

H. Exception Handling pada File

Exception handling pada file merupakan mekanisme penting yang digunakan untuk menangani kesalahan yang mungkin terjadi saat program melakukan operasi file. Kesalahan seperti file tidak ditemukan, izin akses ditolak, file sedang digunakan oleh proses lain, atau kegagalan membaca/menulis dapat menyebabkan program berhenti tiba-tiba jika tidak ditangani dengan baik. Dengan menerapkan exception handling, program dapat tetap berjalan secara aman dan memberikan pesan kesalahan yang lebih jelas kepada pengguna.

```
    string write(string text) {
        ofstream file(filename);
        if (!file) return "Error: Tidak bisa menulis
file.";
        file << text;
        return "Berhasil menulis file.";
    }
};

// Penggunaan
// FileHandler fh("data.txt");
// cout << fh.write("Hello C++") << endl;
// cout << fh.read() << endl;
```

4. PHP – Class OOP untuk operasi file

```
<?php
class FileHandler {
    private $filename;

    public function __construct($filename) {
        $this->filename = $filename;
    }

    public function read() {
        if (!file_exists($this->filename)) {
            return "Error: File tidak ditemukan.";
        }
        return file_get_contents($this->filename);
    }

    public function write($text) {
        $result = file_put_contents($this->filename,
$text);
        return $result !== false ? "Berhasil menulis
file." : "Error menulis file.";
```

```
    }
}

// Penggunaan
// $fh = new FileHandler("data.txt");
// echo $fh->write("Hello PHP");
// echo $fh->read();
?>
```

J. Library / Modul File Handling

Berbagai bahasa pemrograman menyediakan library atau modul khusus yang mempermudah proses file handling. Modul-modul ini membantu program mengakses file, membaca dan menulis data, memanipulasi direktori, serta menangani operasi lanjutan seperti penyalinan, pemindahan, hingga pengelolaan metadata. Dengan memanfaatkan library resmi dari masing-masing bahasa, programmer dapat mengelola file dengan lebih efisien dan menghindari penulisan kode yang berulang.

Setiap bahasa memiliki ekosistem library yang kuat, baik bawaan (built-in) maupun eksternal, yang dirancang untuk menangani operasi file secara aman dan konsisten. Berikut ringkasan library penting terkait file handling pada Python, Java, C++, dan PHP.

1. Python

Python memiliki banyak modul bawaan yang powerful untuk pengelolaan file:

- `os` – operasi dasar seperti cek `file`, `rename`, `delete`, membuat folder.
- `shutil` – operasi tingkat tinggi: `copy`, `move`, `rmtree`.
- `pathlib` – manipulasi path berorientasi objek, lebih modern daripada os.path.
- `io` – kelas untuk operasi stream tingkat rendah.
- `csv, json, pickle` – bekerja dengan format file khusus.

Contoh:

```
from pathlib import Path

file = Path("data.txt")
print(file.exists())
```

2. Java

Java menyediakan library standar yang sangat kaya untuk operasi file:

- `java.io` – membaca/menulis `file`, `stream`, `FileReader`, `FileWriter`.
- `java.nio.file` – API modern untuk operasi file dan direktori (`copy`, `move`, `walk`).
- File dan Files – kelas inti untuk memanipulasi file.

Contoh:

```
import java.nio.file.*;

Files.copy(Paths.get("a.txt"), Paths.get("b.txt"));
```

3. C++

Sejak C++17, pengelolaan file menjadi lebih mudah berkat library standar baru:

- `<fstream>` – membaca dan menulis file (`ifstream`, `ofstream`).
- `<filesystem>` – operasi direktori dan file modern seperti `exists()`, `copy()`, `rename()`.

Contoh:

```
#include <filesystem>
namespace fs = std::filesystem;

cout << fs::exists("data.txt");
```

4. PHP

PHP menyediakan fungsi bawaan yang lengkap untuk file handling tanpa perlu library eksternal:

- `file_exists()`, `unlink()`, `rename()`, `mkdir()` – operasi dasar.
- `file_get_contents()`, `file_put_contents()` – membaca/menulis file secara sederhana.
- `fopen()`, `fread()`, `fwrite()` – kontrol lebih detail menggunakan stream.
- `scandir()` – membaca isi direktori.

Contoh:

```
echo file_exists("data.txt");
```

K. File Buffering dan Stream

File buffering dan stream merupakan dua konsep penting dalam proses input/output (I/O) pada pemrograman. Stream adalah aliran data yang bergerak dari satu titik ke titik lain—misalnya dari file ke program (input stream) atau dari program ke file (output stream). Stream memungkinkan data ditransfer secara bertahap tanpa harus memuat seluruh file ke memori, sehingga proses I/O menjadi lebih fleksibel dan efisien.

Buffering adalah teknik penyimpanan sementara data dalam sebuah area memori yang disebut buffer. Ketika program membaca atau menulis file, data tidak selalu diproses secara langsung byte per byte. Sebaliknya, data dikumpulkan terlebih dahulu dalam buffer, kemudian dikirim sekaligus. Teknik ini mempercepat kinerja I/O karena mengurangi jumlah interaksi langsung dengan perangkat penyimpanan yang relatif lambat.

Dalam berbagai bahasa pemrograman, konsep buffering diimplementasikan melalui kelas atau fungsi khusus yang menangani proses membaca (buffered input) dan menulis (buffered output). Pendekatan ini tidak hanya meningkatkan efisiensi, tetapi juga membuat

alur kerja I/O lebih stabil dan terstruktur. Berikut contoh implementasinya.

1. Java: Contoh BufferedReader & BufferedWriter

Java menyediakan kelas buffering bawaan yang sangat umum digunakan.

BufferedReader (membaca file)

```
import java.io.*;

public class ReadBuffer {
    public static void main(String[] args) {
        try {
            BufferedReader br = new
BufferedReader(new FileReader("data.txt"));
            String line = br.readLine();
            System.out.println(line);
            br.close();
        } catch (Exception e) {
            System.out.println("Error membaca file: "
+ e.getMessage());
        }
    }
}
```

BufferedWriter (menulis file)

```
import java.io.*;

public class WriteBuffer {
    public static void main(String[] args) {
        try {
            BufferedWriter bw = new
BufferedWriter(new FileWriter("data.txt"));
            bw.write("Contoh penulisan menggunakan
buffer.");
            bw.close();
        } catch (Exception e) {
            System.out.println("Error menulis file: "
+ e.getMessage());
        }
    }
}
```

```
    std::filesystem::path file = folder /
"aktivitas.txt";

    std::ofstream out(file.string(),
std::ios::binary);
    // ios::binary mencegah konversi \n ke \r\n di
Windows

    out << "Baris pertama\n";
    out << "Baris kedua\n";
    out.close();

    std::cout << "File tersimpan di: " << file <<
std::endl;
    return 0;
}
```

Catatan:

- `std::ios::binary` mencegah Windows mengubah `\n` menjadi `\r\n`.
- Encoding harus diatur manual (UTF-8), misalnya menulis byte langsung.

d. PHP

PHP menangani newline tergantung pada string yang diberikan, tidak otomatis mengonversi berdasarkan OS.

```
<?php
$folder = "logs";

// Membuat folder
if (!is_dir($folder)) {
    mkdir($folder, 0777, true);
}

$file = $folder . DIRECTORY_SEPARATOR .
"aktivitas.txt";

// Menulis dengan newline \n dan encoding UTF-8
$teks  = "Baris pertama\n";
$teks .= "Baris kedua UTF-8\n";
```

```
file_put_contents($file, $teks);

echo "File tersimpan di: " . realpath($file);
?>
```

Catatan:

- Gunakan `DIRECTORY_SEPARATOR` agar path sesuai OS:
  Windows → `\`
  Linux/Mac → `/`
- PHP tidak mengubah newline: jika Anda tulis `\n`, maka *tetap* `\n`.

M. Studi Kasus Implementasi

Studi kasus berikut menggambarkan bagaimana konsep file handling diterapkan dalam program nyata. Salah satu contoh yang umum dijumpai dalam pengembangan aplikasi adalah program pencatatan log aktivitas pengguna. Program ini digunakan untuk merekam setiap tindakan penting yang dilakukan pengguna, seperti login, akses fitur tertentu, maupun perubahan data. Log aktivitas membantu proses debugging, audit keamanan, dan pemantauan penggunaan sistem. Selain itu, keberadaan file log memungkinkan pengembang menelusuri jejak penggunaan aplikasi ketika terjadi kesalahan, memonitor pola interaksi pengguna, serta menyediakan dokumentasi historis yang dapat digunakan untuk analisis lebih lanjut.

Implementasi berikut disajikan dalam empat bahasa pemrograman—Python, PHP, Java, dan C++—untuk memperlihatkan pendekatan file handling pada berbagai platform pemrograman. Setiap contoh menunjukkan cara membuka, menulis, dan menutup file log sesuai mekanisme masing-masing bahasa, sehingga pembaca dapat memahami perbedaan dan keseragaman konsep yang mendasari proses pengelolaan file pada berbagai ekosistem pengembangan perangkat lunak.

1. Python: Mencatat Aktivitas ke File Log

Python menyediakan mekanisme sederhana untuk bekerja dengan file, termasuk mode append `(a)` untuk menambahkan data tanpa menghapus isi sebelumnya.

Contoh Program:

```
import datetime

def tulis_log(aksi):
    waktu = datetime.datetime.now().strftime("%Y-%m-
%d %H:%M:%S")
    with open("log_aktivitas.txt", "a",
encoding="utf-8") as file:
        file.write(f"{waktu} - {aksi}\n")

# Contoh penggunaan
tulis_log("Pengguna login ke sistem")
tulis_log("Pengguna membuka menu laporan")
```

2. C++: Penulisan Log Menggunakan fstream

Bahasa C++ memanfaatkan library `<fstream>` untuk menulis dan membaca file. Contoh berikut menunjukkan penggunaan mode append `(ios::app)`.

Contoh Program:

```
#include <iostream>
#include <fstream>
#include <ctime>

void tulisLog(const std::string& aksi) {
    std::ofstream file("log_aktivitas.txt",
std::ios::app);
    if (file.is_open()) {
        time_t now = time(0);
        char* dt = ctime(&now);
        dt[strcspn(dt, "\n")] = '\0'; // Hilangkan
newline bawaan ctime

        file << dt << " - " << aksi << std::endl;
    }
}
```

```
int main() {
    tulisLog("User membuka aplikasi");
    tulisLog("User menambahkan data baru");
    return 0;
}
```

3. Java: Implementasi Logging Menggunakan FileWriter/BufferedWriter

Java menggunakan konsep stream dan buffering untuk efisiensi saat menulis file. Contoh berikut menambahkan aktivitas ke dalam file log dengan menggunakan `BufferedWriter`.

Contoh Program:

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class LogAktivitas {
    public static void tulisLog(String aksi) {
        try (BufferedWriter bw = new
BufferedWriter(new FileWriter("log_aktivitas.txt",
true))) {
            String waktu =
LocalDateTime.now().format(DateTimeFormatter.ofPatter
n("yyyy-MM-dd HH:mm:ss"));
            bw.write(waktu + " - " + aksi);
            bw.newLine();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        tulisLog("User login");
        tulisLog("User menghapus data");
    }
}
```

4. PHP: Merekam Aktivitas pada File Log Server

PHP sering digunakan pada aplikasi web, sehingga pencatatan log sangat berguna untuk memantau aktivitas pengguna melalui browser.

Contoh Program:

```
<?php
function tulis_log($aksi) {
    $waktu = date("Y-m-d H:i:s");
    $baris = $waktu . " - " . $aksi . PHP_EOL;
    file_put_contents("log_aktivitas.txt", $baris,
FILE_APPEND);
}

// Contoh penggunaan
tulis_log("User mengakses halaman dashboard");
tulis_log("User mengupdate profil");
?>
```

N. Best Practices dalam File Handling

Agar pengelolaan file berjalan aman, efisien, dan bebas dari kesalahan, beberapa praktik terbaik perlu diterapkan dalam proses pengembangan perangkat lunak. Praktik-praktik ini membantu mencegah korupsi data, kehilangan file, kerentanan keamanan, maupun kebocoran sumber daya sistem. Berikut tujuh poin yang direkomendasikan dalam implementasi file handling pada berbagai bahasa dan platform.

1. Selalu Menutup File Setelah Digunakan

Menutup file merupakan langkah penting agar sumber daya sistem (file descriptor/handle) tidak bocor.

- Gunakan mekanisme otomatis seperti `with` (Python), `try-with-resources` (Java), RAII (C++), dan `fclose()` yang konsisten pada PHP/C.
- File yang tidak ditutup berpotensi menyebabkan error *file locked* dan kehilangan data akibat buffer yang belum flush.

2. Lakukan Validasi Path dan Nama File

Validasi diperlukan untuk mencegah akses ke lokasi yang tidak aman atau file yang tidak diizinkan.

- Pastikan path tidak mengandung karakter ilegal.
- Cegah path traversal attack seperti `../../etc/passwd`.
- Gunakan fungsi path standar (`Pathlib`, `Paths.get()`, `std::filesystem`, `realpath()`).

3. Terapkan Exception Handling untuk Mengantisipasi Error

Pengelolaan error membantu program tetap berjalan normal meskipun terjadi kegagalan seperti:

- File tidak ditemukan
- Akses ditolak
- Ruang penyimpanan penuh
- Format atau encoding tidak sesuai
  Gunakan blok `try-except`, `try-catch`, atau pengecekan error khusus sesuai bahasa.

4. Gunakan Buffering dan Stream untuk Efisiensi

File berukuran besar sebaiknya tidak dibaca sekaligus ke memori. Buffering dan streaming membuat proses I/O lebih cepat dan hemat memori.

- Contoh: `BufferedReader/BufferedWriter` (Java), `buffered I/O` (Python), `std::ifstream` (C++).

5. Buat Salinan Cadangan (Backup) Secara Berkala

Backup diperlukan untuk menjaga integritas data terutama pada aplikasi yang sering menulis file log, konfigurasi, atau data kritis.

- Terapkan mekanisme rolling backup, timestamped backup, atau auto-rotation.
- Simpan backup di folder terpisah atau lokasi aman.

6. Perhatikan Encoding dan Format Data

Perbedaan encoding dapat menyebabkan karakter salah tampil atau file korup.

- Gunakan encoding konsisten, misalnya UTF-8.
- Tentukan newline (`\n`, `\r\n`) sesuai OS jika diperlukan.
- Dokumentasikan format file sehingga dapat dibaca oleh aplikasi lain.

7. Hindari Mengakses File Berulang Tanpa Kebutuhan

Akses file yang tidak efisien menyebabkan performa buruk.

- Simpan hasil pembacaan sementara (caching) jika diperlukan berkali-kali.
- Satukan operasi penulisan daripada menulis satu per satu dalam loop kecil.
- Tutup file segera setelah operasi selesai.

BAB 13
MODULARISASI & LIBRARY

A. Pengantar Modularisasi

Modularisasi adalah teknik dalam rekayasa perangkat lunak untuk memecah program besar menjadi bagian-bagian kecil (modul) yang lebih mudah dikelola. Setiap modul biasanya memiliki tanggung jawab tunggal, memudahkan pengembangan, pengujian, pemeliharaan, dan reuse.

Manfaat Modularisasi :

1. Meningkatkan keterbacaan kode – modul yang ringkas lebih mudah dipahami.
2. Mendukung reuse (penggunaan ulang) – modul yang sama bisa digunakan di proyek lain.
3. Memudahkan debugging – kesalahan dapat ditelusuri ke modul tertentu.
4. Mendukung kerja tim – beberapa programmer bisa mengerjakan modul berbeda secara paralel.

B. Konsep Modul dan Fungsi

Modul dan fungsi merupakan dua komponen penting dalam pemrograman yang berperan dalam membangun struktur program yang lebih terorganisasi, efisien, dan mudah dipelihara (Rahmawati, 2024). Keduanya memungkinkan programmer memecah program besar menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola.

2. C++ — Header dan Implementasi

Struktur Folder:

```
proyek/
 ├─ modul_matematika/
 │   ├─ luas.h
 │   └─ luas.cpp
 └─ main.cpp
```

modul_matematika/luas.h

```
#ifndef LUAS_H
#define LUAS_H

int luasPersegi(int sisi);

#endif
```

modul_matematika/luas.cpp

```
#include "luas.h"

int luasPersegi(int sisi) {
    return sisi * sisi;
}
```

main.cpp

```
#include <iostream>
#include "modul_matematika/luas.h"

int main() {
    std::cout << luasPersegi(6);
    return 0;
}
```

3. Java — Package dan Import

Struktur Folder:

```
proyek/
 ├─ modul_matematika/
 │   └─ Luas.java
 └─ Main.java
```

modul_matematika/Luas.java

```
package modul_matematika;

public class Luas {
    public static int luasPersegi(int sisi) {
        return sisi * sisi;
    }
}
```

Main.java

```
import modul_matematika.Luas;

public class Main {
    public static void main(String[] args) {
        System.out.println(Luas.luasPersegi(6));
    }
}
```

4. PHP — include / require

Struktur Folder:

```
proyek/
 ├─ modul_matematika/
 │    └─ luas.php
 └─ main.php
```

modul_matematika/luas.php

```
<?php
function luas_persegi($sisi) {
    return $sisi * $sisi;
}
?>
```

main.php

```
<?php
include 'modul_matematika/luas.php';

echo luas_persegi(6);
?>
```

D. Library (Pustaka) dalam Pemrograman

Library adalah sekumpulan modul, fungsi, atau kelas yang telah dikemas dan siap digunakan untuk menyelesaikan tugas tertentu tanpa harus menulis ulang kode dari awal. Library dibuat untuk mempercepat proses pengembangan, meningkatkan konsistensi, dan menyediakan solusi standar yang telah teruji.

1. Fungsi dan Peran Library

Library berfungsi menyediakan kode siap pakai yang telah dioptimalkan untuk berbagai kebutuhan. Beberapa peran penting library antara lain:

- Menyediakan fitur siap pakai seperti manipulasi string, matematika, grafik, jaringan, file handling, dan database.
- Mengurangi duplikasi pekerjaan, karena programmer tidak perlu membangun semuanya dari nol.
- Meningkatkan kualitas dan keamanan, karena library populer biasanya diuji oleh komunitas luas.
- Menstandarkan cara kerja, sehingga kolaborasi lebih mudah.

2. Jenis-Jenis Library

Library dapat dikategorikan menjadi beberapa kelompok:

- Library Standar (Standard Library)
  Sudah disediakan oleh bahasa pemrograman (misalnya: math di Python, java.util di Java).
- Library Eksternal (Third-Party Library)
  Dibuat oleh komunitas atau vendor tertentu, dan dipasang secara terpisah.
- Framework
  Struktur program lengkap yang menyediakan aturan, pola, dan library di dalamnya (misalnya Django, Laravel, React).
- Package / Modul Tingkat Tinggi
  Kumpulan library dalam satu distribusi (misalnya NumPy, Boost, Laravel Collection).

E. Penggunaan Library Standar

Hampir semua bahasa pemrograman menyediakan *Standard Library* yang berisi kumpulan fungsi, kelas, dan modul siap pakai untuk membantu menyelesaikan berbagai tugas dasar—mulai dari operasi matematis, manipulasi string, pengelolaan file, hingga pemrosesan data. Berikut contoh penggunaan library standar dari beberapa bahasa:

1. Python – Library `math`

```
import math

print(math.sqrt(49))
print(math.pi)
```

2. Java – Library `java.util`

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<Integer> data = new ArrayList<>();
        data.add(10);
        data.add(20);
        System.out.println(data);
    }
}
```

3. C++ – Library `<vector>`

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> angka = {1, 2, 3};
    angka.push_back(4);

    for (int a : angka)
        cout << a << " ";

    return 0;
}
```

4. PHP – Fungsi Built-in

```
<?php
echo strlen("Hello World");
echo strtoupper("php modular");
?>
```

F. Penggunaan Library Eksternal

Library eksternal memberikan kemampuan yang jauh lebih luas dibandingkan fasilitas bawaan bahasa pemrograman. Melalui library ini, pengembang dapat menambahkan fitur khusus—seperti koneksi database, pengolahan data tingkat lanjut, pembuatan antarmuka grafis, hingga pengembangan aplikasi berbasis web—tanpa harus menulis semuanya dari awal. Dengan demikian, library eksternal mempercepat proses pengembangan dan memperkaya fungsionalitas aplikasi.

1. Python – Menggunakan `requests`

```
import requests
response = requests.get("https://example.com")
print(response.status_code)
```

2. Java – Menggunakan Gson (Google JSON Library)

a. Tambahkan dependensi (Maven):

```
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.10.1</version>
</dependency>
```

b. Pemakaian:

```
import com.google.gson.Gson;

public class Main {
    public static void main(String[] args) {
        Gson gson = new Gson();
        String json = gson.toJson(new int[]{1,2,3});
        System.out.println(json);
    }
}
```

3. C++ – Menggunakan Library Boost

```
#include <boost/algorithm/string.hpp>
#include <iostream>
#include <string>
using namespace std;

int main() {
    string teks = "Hello World";
    boost::to_upper(teks);
    cout << teks;
}
```

4. PHP – Menggunakan Library Eksternal via Composer

a. Install library (misal Guzzle):

```
composer require guzzlehttp/guzzle
```

b. Penggunaan:

```
<?php
require 'vendor/autoload.php';

use GuzzleHttp\Client;

$client = new Client();
$res = $client->get('https://example.com');

echo $res->getStatusCode();
?>
```

G. Modularisasi Proyek Besar

Struktur proyek besar memisahkan kode berdasarkan tanggung jawab.

Konsep

1) Modul input/output
2) Modul logika / bisnis
3) Modul utilitas / helper
4) Paket eksternal

Diagram Struktur Proyek

```
Proyek Utama/
 ├─ main.py
 ├─ modul_input/
 ├─ modul_logika/
 ├─ modul_utilitas/
 └─ lib_eksternal/
```

Contoh Python

```
# modul_logika/hitung_rata.py
def rata_rata(list_nilai):
    return sum(list_nilai)/len(list_nilai)

# main.py
from modul_logika.hitung_rata import rata_rata
nilai = [80,90,70]
print(rata_rata(nilai))  # 80.0
```

Contoh Java

```
package modul_logika;
public class Hitung { public static double
rataRata(int[] arr){...} }
import modul_logika.Hitung;
System.out.println(Hitung.rataRata(new
int[]{80,90,70}));
```

Contoh PHP

```
include 'modul_logika/hitung_rata.php';
$nilai=[80,90,70];
echo rata_rata($nilai);
```

Contoh C++

Struktur folder: `modul_logika/hitung_rata.h/cpp` `main.cpp` memanggil fungsi `rata_rata`.

BAB 14
STUDI KASUS MINI PROJECT

A. Mini Project 1 — Aplikasi Kalkulator

1. Konsep

Aplikasi kalkulator adalah contoh klasik untuk menggabungkan konsep:

- Fungsi & modularisasi
- Input–proses–output
- Penggunaan operasi aritmatika

Setiap bahasa menampilkan implementasi yang modular.

2. PYTHON — Kalkulator

kalkulator.py

```
def tambah(a, b): return a + b
def kurang(a, b): return a - b
def kali(a, b): return a * b
def bagi(a, b): return a / b
```

main.py

```
import kalkulator as k

a = float(input("Angka 1: "))
b = float(input("Angka 2: "))
op = input("Operasi (+ - * /): ")

opsi = {"+": k.tambah, "-": k.kurang, "*": k.kali,
"/": k.bagi}
print("Hasil =", opsi[op](a, b))
```

3. C++ — Kalkulator

kalkulator.h

```
double tambah(double a, double b);
double kurang(double a, double b);
double kali(double a, double b);
double bagi(double a, double b);
```

kalkulator.cpp

```
#include "kalkulator.h"

double tambah(double a, double b){ return a+b; }
double kurang(double a, double b){ return a-b; }
double kali(double a, double b){ return a*b; }
double bagi(double a, double b){ return a/b; }
```

main.cpp

```
#include <iostream>
#include "kalkulator.h"
using namespace std;

int main(){
    double a,b;
    char op;

    cout<<"Angka 1: "; cin>>a;
    cout<<"Angka 2: "; cin>>b;
    cout<<"Operasi (+ - * /): "; cin>>op;

    switch(op){
        case '+': cout<<tambah(a,b); break;
        case '-': cout<<kurang(a,b); break;
        case '*': cout<<kali(a,b); break;
        case '/': cout<<bagi(a,b); break;
    }
}
```

4. JAVA — Kalkulator

Calc.java

```
public class Calc {
    public static double tambah(double a, double b){
return a+b; }
    public static double kurang(double a, double b){
return a-b; }
    public static double kali(double a, double b){
return a*b; }
    public static double bagi(double a, double b){
return a/b; }
}
```

Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        double a, b; String op;

        System.out.print("Angka 1: "); a =
in.nextDouble();
        System.out.print("Angka 2: "); b =
in.nextDouble();
        System.out.print("Operasi (+ - * /): "); op =
in.next();

        switch(op){
            case "+":
System.out.println(Calc.tambah(a,b)); break;
            case "-":
System.out.println(Calc.kurang(a,b)); break;
            case "*":
System.out.println(Calc.kali(a,b)); break;
            case "/":
System.out.println(Calc.bagi(a,b)); break;
        }
    }
}
```

4. JAVA — Data Mahasiswa

```
import java.util.*;

class Mhs {
    String nama, nim;
    Mhs(String n, String i){ nama=n; nim=i; }
}

public class Main {
    static ArrayList<Mhs> data = new ArrayList<>();

    static void tambah(){
        Scanner in = new Scanner(System.in);
        System.out.print("Nama: ");
        String nama = in.nextLine();
        System.out.print("NIM : ");
        String nim = in.nextLine();
        data.add(new Mhs(nama,nim));
    }

    static void tampil(){
        for(Mhs m : data)
            System.out.println(m.nim + " - " +
m.nama);
    }

    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int p;
        do{
            System.out.print("1.Tambah 2.Tampil
3.Keluar: ");
            p = in.nextInt();
            if(p==1) tambah();
            else if(p==2) tampil();
        } while(p!=3);
    }
}
```

5. PHP — Data Mahasiswa

```
<?php
$mahasiswa = [];

function tambah(&$mhs){
    $nama = readline("Nama: ");
    $nim  = readline("NIM : ");
    $mhs[] = ["nama"=>$nama, "nim"=>$nim];
}

function tampil($mhs){
    foreach($mhs as $m){
        echo $m["nim"]." - ".$m["nama"].PHP_EOL;
    }
}

while(true){
    echo "1.Tambah 2.Tampil 3.Keluar: ";
    $p = readline();

    if($p == "1") tambah($mahasiswa);
    else if($p == "2") tampil($mahasiswa);
    else break;
}
?>
```

DAFTAR PUSTAKA

Aldi, M., Islam, U., Sumatera, N., Aji, M. I., Islam, U., Sumatera, N., & Serdang, D. (2025). *PERANCANGAN SISTEM DIGITAL SURAT MASUK DAN SURAT*. *2*(2), 65–77.

Andaria, A. C. (2025). *PENGENALAN BAHASA PEMPROGRAMAN UNTUK PEMULA*. PT. Hadla Media Informasi. Jawa Barat.

Andre – Admin Duniailkom.com. (2025). *duniailkom*. Dunia Ilmu Komputer. https://www.duniailkom.com/

Anggraini, A., Rahelta, C. E., Simangunsong, M. A., & Harahap, Z. (2025). *PERANCANGAN APLIKASI SISTEM KEUANGAN BERBASIS DEKSTOP IMPLEMENTATION OF OBJEC ORIENTED PROGRAMMING IN DESIGNING DESKTOP-BASED FINANCIAL SYSTEM APPLICATIONS*. *April*, 7311–7316.

Avorizano, A., Fikri, A., Mujirudin, M., Aji, R. S., Anugrah, R. D., Intia, B. C., Hanif, C., & Rusneli, R. A. (2025). Mengenalkan Keahlian Pemrograman Siswa SMPN Satu Atap 01 Pulau Pari dengan Memanfaatkan Ilmu Logika. *Jurnal Pengabdian Kepada Masyarakat Nusantara*, *6*(1), 619–624. https://doi.org/10.55338/jpkmn.v6i1.3101

Cahyaningtyas, C., & Gudiato, C. (2025). *Meningkatkan Logika Pemrograman Siswa SMK Melalui Pelatihan Pengenalan Flowchart dan Pseudocode*. *4*(2), 1534–1539. https://doi.org/https://doi.org/10.58266/jpmb.v4i2.673

Candra, A. P. (2025). *Analisis Data Menggunakan Python : Memperkenalkan Pandas dan NumPy*. *3*(1), 11–16. https://doi.org/https://doi.org/10.62386/jised.v3i1.118

Dafit Afianto, S. K. M. K., & Riko Muhammad Suri, S. K. M. K. (2025). *Pengantar Sistem Informasi, Pendekatan Praktis untuk Pemula*. PT. Global Mitra Mediatama. https://books.google.co.id/books?id=poiHEQAAQBAJ

De Kweldju, A. (2023). *Buku Ajar Pengantar Ilmu Komputer*. Penerbit NEM. https://books.google.co.id/books?id=OmzFEAAAQBAJ

Ikhsan, F. K., Hasan, F. N., Afandi, I. R., Fathurrohman, S., Salma, T. D., Hafsari, R., Nasution, E., Ismanto, E., Wiwit Supriyanti, S. K. M.

K., & others. (2025). *Struktur Data dengan Bahasa Java*. PT Penerbit Qriset Indonesia. https://books.google.co.id/books?id=lIqREQAAQBAJ

Komalasari, R., Widians, J. A., Meilani, B. D., Arifin, N. Y., Sepriano, S., Syam, S., Mahendra, G. S., Rachman, A., Muryanah, S., Samsumar, L. D., & others. (2023). *PENGANTAR ILMU KOMPUTER : TEORI KOMPREHENSIF PERKEMBANGAN ILMU KOMPUTER TERKINI*. PT. Sonpedia Publishing Indonesia. https://books.google.co.id/books?id=znHVEAAAQBAJ

Kurniawan, D., & Publisher, A. R. (2025). *Mengenal Profesi Software Engineer*. Afdan Rojabi Publisher. https://books.google.co.id/books?id=TFuVEQAAQBAJ

Mandowen, S. A., Rustiyana, H., Nistrina, K., Prayudani, S., Judijanto, L., Efitra, E., & Putri, N. O. (2025). *Dasar Pemrograman : Teori \& Aplikasi*. PT. Sonpedia Publishing Indonesia. https://books.google.co.id/books?id=c8qREQAAQBAJ

Manuaba, I. B. K., Purnawati, N. W., Arsana, I. N. A., Putri, N. A. R., Phan, I. K., Putra, P. M., Sindu, I. G. P., irviantina, S., Agusdi, Y., & Pamela, E. (2024). *Buku Ajar Pengantar Teknologi Informasi*. PT. Sonpedia Publishing Indonesia. https://books.google.co.id/books?id=-zAxEQAAQBAJ

Maskhuliah, P., Selayar, T. M., Putri, A. S., Agama, I., Negeri, I., Muluk, F., Putih, J. M., Waena, B., Jayapura, K., & Indonesia, P. (2025). *Proposisi dalam Matematika : Pemahaman Tentang Negasi Konjungsi , dan Disjungsi dan Penerapan Operasi Logika dalam Analisis Matematika*. *September*. https://doi.org/https://doi.org/10.62383/algoritma.v3i5.744

Monica, G., & Candradewini, C. (2025). *PELATIHAN BERBASIS KOMPETENSI JUNIOR WEB PROGRAMMER DI DINAS TENAGA KERJA KOTA CIREBON*. *16*(2), 152–161. https://doi.org/https://doi.org/10.24198/jane.v16i2.61296

Nurhadi, N., Jatmiko, A. R., Legito, L., Saputra, E. A., Syafa'at, F., Surianto, D. F., Komalasari, R., Mukhlis, I. R., Sulistyowati, S., & others. (2023). *BUKU AJAR LOGIKA & ALGORITMA*. PT. Sonpedia Publishing Indonesia. https://books.google.co.id/books?id=n2XOEAAAQBAJ

Octarina, S., Puspita, F. M., Yuliza, E., & Meeting, G. (2025). *Jurnal*

Pepadu Jurnal Pepadu. *6*(1), 56–66. https://doi.org/https://doi.org/10.29303/pepadu.v6i1.6457

Panggabean, S., Gata, W., Syarif, A. R., & Rahmadani, S. (2021). *Implementasi Algoritma Dijkstra Untuk Menentukan Jalur Terpendek Wilayah Pasar Minggu Dan STMIK Nusamandiri Jakarta*. *9*(1), 78–84. https://doi.org/https://doi.org/10.31294/swabumi.v9i1.9574

Panggabean, S., Saputro, W. J., Zakiy, F. W., Lestari, T., Rifqi, A., & Technology, I. (2023). *DARUNNAJAH VOTE SYSTEM APPLICATION DESIGN USING PHP*. 25–38. https://doi.org/https://doi.org/10.35585/inspir.v13i2.61

Prastyo, D., Mursyidin, I. H., & Irawan, D. (2025). *Sistem Informasi Terpusat untuk Manajemen Dokumen , Penelitian , dan*. *7*(3). https://doi.org/10.32877/bt.v7i3.2182

Rahmadi Asri, S. I. M. K., Ully Muzakir, M. T., Priska Choirina, M. T. T., Yazid Aufar, M. K., Dian Nugraha, S. S. T. M. I. T., Ira Zulfa, S. T. M. C., Devi Chrisman Lase, S. K. M. K., Syukriadi, S. K. M. K., Mera Kartika Delimayanti, S. S. M. T., Agus Nur Khomarudin, M. K., & others. (2025). *Pengantar Teknik Informatika*. Azzia Karya Bersama. https://books.google.co.id/books?id=b7JbEQAAQBAJ

Rahmawati, L. S. (2024). Desain Pengembangan Website dengan Arsitektur Model View Controller pada Framework Laravel. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *6*(4), 785–790. https://doi.org/https://doi.org/10.47233/jteksis.v6i4.1497

Richki Hardi, S. T. M. E. (2021). *Arsitektur dan Organisasi Komputer*. uwais inspirasi indonesia. https://books.google.co.id/books?id=0tZUEAAAQBAJ

Saputra, H., Hutapea, G., & Ahmad, F. A. (2025). *Pengembangan Aplikasi E-Commerce Multi Product Pada Usaha Mikro Kecil Menengah Berbasis Web*. *3*(4), 1837–1842. https://doi.org/https://doi.org/10.31004/jerkin.v3i4.775

Sulianta, F. (2025). *Buku Algoritma dan Pemrograman*. Researchgate.net.

Sulistyowati, S., Syam, S., Skawanti, J. R., Arfyanti, I., Febriana, R. W., Meilani, B. D., Dedih, D., Zainuddin, N., Efitra, E., & Safitri, N. (2025). *Buku Ajar Pemrograman Dasar*. PT. Green Pustaka Indonesia.

H

Hash Function – Fungsi hash unik.
HashMap – Struktur key–value cepat (Java).
Header File – Deklarasi fungsi C++.
Heap Memory – Penyimpanan dinamis.
Hexadecimal – Sistem bilangan basis 16.
High Availability – Sistem uptime tinggi.
Hotfix Deployment – Perbaikan darurat instan.
HTML – Bahasa markup web.
HTTP/HTTPS – Protokol komunikasi web.
Hybrid App – Gabungan web dan native.
Hybrid Encryption – Kombinasi simetris & asimetris.
Hypervisor – Teknologi virtualisasi server.

I

IDE – Editor+compiler dalam satu paket.
IDE Plugin – Ekstensi fitur IDE.
If Statement – Percabangan kondisi.
Indexing – Akselerasi pencarian data.
Index Scan – Akses query lewat indeks.
Immutable – Data tidak dapat berubah.
Immutable Object – Objek non-mutable.
Import – Memanggil modul/library.
Increment/Decrement – ++, -- operasi kenaikan/penurunan.
Inline Function – Fungsi dieksekusi langsung.
Infrastructure as Code – Server didefinisi via script.
Input – Data masuk.
Input Sanitization – Penyaringan input berbahaya.
Instance – Objek hasil dari class.
Instance Variable – Variabel terikat objek.
Integer – Bilangan bulat.
Interpreter – Eksekusi kode per-baris.

Procedure – Fungsi tanpa return.
Prototype – Versi konsep awal.
Pseudo Code – Algoritma ringkas.

Q

Query Optimization – Optimasi query SQL.
Query String – Parameter dalam URL.
Queue – Struktur FIFO.
Queue Worker – Pemrosesan antrean background.
Quantum Computing – Komputasi qubit.

R

Recursion – Fungsi memanggil diri sendiri.
Refactoring – Perbaikan kode tanpa ubah fungsinya.
Regression Testing – Tes ulang setelah update.
Repository – Tempat penyimpanan kode.
REST API – Endpoint HTTP web.
Reverse Proxy – Server perantara request.
Rollback – Kembali ke versi lama.
Routing Table – Peta rute jaringan.
Runtime – Waktu program berjalan.
Runtime Error – Error saat run.

S

Salting Password – Penambahan acak sebelum hashing.
Schema Database – Struktur tabel database.
Scrum Methodology – Kerangka kerja Agile.
SDK – Toolkit pembuatan sistem.
Semaphore – Pengatur thread concurrent.
Serialization – Objek → byte untuk kirim/simpan.
Serverless Architecture – Eksekusi tanpa kelola server.
Shell Script – Automasi perintah terminal.
Singleton Pattern – Hanya satu instance objek per aplikasi.
Source Code – Kode sumber asli.

SSL Certificate – Sertifikat keamanan.
Stack Memory – Memori fungsi stack.
Static Typing – Tipe ditentukan compile-time.
String – Kumpulan karakter.
Switch-Case – Percabangan banyak kondisi.
Syntax – Tata bahasa program.
Syntax Error – Salah penulisan kode.

T
Thread – Jalur eksekusi paralel.
Token – Unit terkecil sintaks.
Try-Catch – Penanganan error runtime.
Tuple – Struktur immutable.

U
Unsigned Integer – Bilangan tanpa tanda negatif.
UTF-8 – Encoding universal.
User Input – Masukan user saat run.

V
Variable – Tempat penyimpanan nilai.
Vector (C++) – List dinamis berbasis memori.
Void – Fungsi tanpa nilai return.

W
Web Programming – Pemrograman berbasis web.
While Loop – Loop berdasarkan kondisi.
Wrapper Class – Pembungkus tipe primitif (Java).

X
XML – Penyimpanan data terstruktur.

Y
YAML – Format data sederhana.

Yield (Python) – Menghasilkan nilai bertahap (generator).

Z

Zero-Based Indexing – Index dimulai dari 0.
Zip (Python) – Menggabung dua list menjadi pair.
Zombie Process – Proses mati tersisa di OS.

BIOGRAFI PENULIS

Supriadi Panggabean lahir di Pandan, Kabupaten Tapanuli Tengah, pada tahun 1993. Penulis menempuh pendidikan Sarjana di bidang Teknik Informatika di Universitas Indraprasta PGRI Jakarta, kemudian melanjutkan studi dan meraih gelar Magister Pendidikan Matematika dan IPA di universitas yang sama, serta gelar Magister Ilmu Komputer dari Universitas Nusa Mandiri Jakarta.

Sejak menempuh pendidikan, penulis telah menunjukkan minat yang besar dalam dunia pendidikan dan teknologi informasi. Penulis mengabdi delapan tahun sebagai tenaga kependidikan dan pengajar di Yayasan Al-Hikmah Mampang Jakarta Selatan (2014–2022), dan sejak 2022 aktif sebagai dosen di Universitas Darunnajah, membimbing mahasiswa dalam mata kuliah terkait komputer dan pemrograman.

Selain pengalaman mengajar, penulis juga tertarik dalam pengembangan literatur pendidikan dan teknologi. Buku ini merupakan karya pertama penulis dalam bentuk buku; sebelumnya, penulis lebih banyak menulis artikel ilmiah yang diterbitkan di jurnal.

Dengan latar belakang pendidikan, pengalaman mengajar, dan ketertarikan dalam dunia teknologi, penulis berharap buku ini dapat menjadi panduan yang bermanfaat bagi pembaca dalam memahami dasar-dasar pemrograman, mengembangkan keterampilan coding, serta memotivasi pembaca untuk terus belajar dan berinovasi di bidang teknologi informasi.

BAHASA PEMROGRAMAN DASAR

MENGGUNAKAN PHYTON, C++, JAVA DAN PHP

Perkembangan teknologi informasi yang semakin pesat menuntut penguasaan keterampilan pemrograman yang tidak hanya berorientasi pada pemahaman teoretis, tetapi juga mampu diaplikasikan secara nyata dalam pemecahan berbagai permasalahan. Pemrograman kini menjadi fondasi utama dalam pengembangan perangkat lunak, sistem informasi, aplikasi web, hingga teknologi cerdas berbasis data. Oleh karena itu, dibutuhkan sumber pembelajaran yang sistematis, mudah dipahami, dan relevan dengan kebutuhan pembelajar pemula maupun mahasiswa di bidang teknologi informasi.

Buku ini dirancang sebagai panduan komprehensif untuk mempelajari dasar-dasar pemrograman secara bertahap dan terstruktur. Materi dimulai dari konsep fundamental seperti variabel, tipe data, operator, percabangan, perulangan, dan fungsi, kemudian dilanjutkan dengan pembahasan modularisasi program serta perancangan program sederhana yang aplikatif. Setiap konsep disajikan secara ringkas, logis, dan disertai contoh kasus sehingga pembaca tidak hanya memahami apa dan bagaimana, tetapi juga mengapa suatu konsep digunakan.

Keunikan buku ini terletak pada penggunaan empat bahasa pemrograman populer, yaitu Python, C++, Java, dan PHP. Pendekatan multibahasa ini bertujuan untuk memberikan wawasan yang lebih luas mengenai perbedaan gaya penulisan, struktur sintaks, serta paradigma pemrograman yang digunakan pada masing-masing bahasa. Dengan demikian, pembaca dapat membangun cara berpikir algoritmik yang lebih fleksibel dan adaptif, sekaligus memiliki bekal untuk memilih dan mendalami bahasa pemrograman yang paling sesuai dengan kebutuhan akademik maupun profesional.

Buku ini sangat cocok digunakan sebagai buku ajar pada mata kuliah pengantar pemrograman, referensi belajar mandiri bagi pemula, serta bahan pendukung bagi siapa saja yang ingin membangun fondasi pemrograman yang kuat dan berkelanjutan di era digital.

Penerbit UDN Press
Ulujami, Jakarta Selatan
press.darunnajah.ac.id
@udn_press
udnpress@darunnajah.ac.id

ISBN 978-634-04-7828-0
9 786340 478280