

**MODUL AJAR DEEP LEARNING**  
**MATA PELAJARAN : INFORMATIKA**  
**BAB 7 : ALGORITMA DAN PEMROGRAMAN**

**A. IDENTITAS MODUL**

**Nama Sekolah** : .....  
**Nama Penyusun** : .....  
**Mata Pelajaran** : **Informatika**  
**Kelas / Fase /Semester** : **X/ E / Ganjil**  
**Alokasi Waktu** : **12 Jam Pelajaran (6 Pertemuan @ 2 JP)**  
**Tahun Pelajaran** : **2024 / 2025**

**B. IDENTIFIKASI KESIAPAN PESERTA DIDIK**

- **Pengetahuan Awal:** Peserta didik diharapkan memiliki pemahaman dasar tentang berpikir komputasional (sebagaimana yang sudah dibahas pada bab sebelumnya). Ini meliputi kemampuan dekomposisi masalah, pengenalan pola, abstraksi, dan merumuskan algoritma sederhana dalam bahasa alami. Mereka mungkin sudah pernah berinteraksi dengan aplikasi atau game yang didasarkan pada logika pemrograman (misalnya, game puzzle logika, aplikasi visual programming seperti Scratch di tingkat SMP). Beberapa mungkin sudah memiliki pengalaman bermain dengan blok coding atau editor teks sederhana.
- **Minat:** Minat terhadap pemrograman sangat bervariasi. Pendekatan yang mengaitkan pemrograman dengan pembuatan aplikasi yang mereka gunakan sehari-hari (game sederhana, kalkulator, chatbot), atau pemecahan masalah yang relevan dengan hobi mereka, akan meningkatkan minat. Penggunaan platform pemrograman visual yang *user-friendly* seperti Scratch atau Blockly (untuk transisi ke teks) dapat membuat pengalaman awal lebih menyenangkan.
- **Latar Belakang:** Peserta didik berasal dari latar belakang yang beragam dalam hal akses dan paparan terhadap teknologi dan pemrograman. Beberapa mungkin sudah memiliki komputer pribadi dan koneksi internet yang stabil, sementara yang lain mungkin terbatas. Beberapa mungkin sudah pernah mencoba *coding* secara otodidak, sementara yang lain sama sekali belum. Perlu pendekatan berdiferensiasi untuk menjembatani kesenjangan ini.
- **Kebutuhan Belajar:**
  - **Visual:** Membutuhkan diagram alur (flowchart), contoh kode dengan *highlighting* sintaks, video tutorial langkah demi langkah, simulasi eksekusi kode, atau visualisasi data struktur.
  - **Auditori:** Membutuhkan penjelasan lisan yang jelas tentang konsep pemrograman, diskusi kelompok untuk memecahkan *bug*, atau mendengarkan penjelasan dari guru/teman.
  - **Kinestetik:** Membutuhkan praktik langsung menulis kode, debugging, mencoba berbagai input, atau bahkan aktivitas *unplugged* untuk mensimulasikan kontrol

aliran program.

- Beberapa peserta didik mungkin memerlukan bimbingan ekstra dalam memahami sintaks atau logika pemrograman yang kompleks, sementara yang lain mungkin siap untuk tantangan proyek yang lebih besar dan mandiri.

### C. KARAKTERISTIK MATERI PELAJARAN

- **Jenis Pengetahuan yang Akan Dicapai:** Pengetahuan konseptual (definisi algoritma, pemrograman, bahasa pemrograman, variabel, tipe data, operator, struktur kontrol/kondisional, perulangan, fungsi), pengetahuan prosedural (menulis algoritma dalam pseudocode/flowchart, menulis kode program sederhana, melakukan debugging, menguji program), dan pengetahuan metakognitif (merencanakan solusi sebelum coding, merefleksikan efisiensi kode, memilih struktur kontrol yang tepat untuk masalah tertentu).
- **Relevansi dengan Kehidupan Nyata Peserta Didik:** Materi ini sangat relevan dan fundamental di era digital. Pemrograman adalah bahasa yang menggerakkan sebagian besar teknologi yang mereka gunakan (ponsel, aplikasi, game, website). Belajar pemrograman mengembangkan kemampuan problem-solving, logika, kreativitas, dan ketelitian yang bermanfaat di berbagai bidang.
- **Tingkat Kesulitan:** Tingkat kesulitan materi ini bersifat moderat hingga tinggi. Memahami konsep dasar variabel dan urutan instruksi relatif mudah. Namun, memahami struktur kontrol yang kompleks (if-else bersarang, perulangan bertingkat), menemukan dan memperbaiki *bug*, serta menulis program yang efisien untuk masalah yang lebih besar memerlukan ketelitian, penalaran logis yang kuat, dan banyak latihan.
- **Struktur Materi:** Materi diawali dengan pengantar dari algoritma ke pemrograman, pengenalan bahasa pemrograman sederhana (disarankan Python atau Blockly/Scratch sebagai jembatan), konsep dasar (variabel, tipe data, operator), struktur kontrol (kondisional dan perulangan), hingga pengenalan fungsi dan *debugging* sederhana. Setiap konsep dibangun secara bertahap.
- **Integrasi Nilai dan Karakter:**
  - **Penalaran Kritis:** Mengembangkan kemampuan menganalisis masalah, merancang solusi logis (algoritma), dan menemukan serta memperbaiki *bug* (kesalahan) dalam program.
  - **Kreativitas:** Mendorong peserta didik untuk merancang program yang inovatif, menemukan berbagai cara untuk memecahkan masalah, dan mengimplementasikan ide-ide baru.
  - **Kolaborasi:** Melatih kemampuan bekerja sama dalam kelompok untuk merancang algoritma, menulis kode, dan melakukan *debugging*.
  - **Kemandirian:** Mendorong ketekunan dan kesabaran dalam menghadapi *bug* dan tantangan pemrograman, serta mencari solusi secara mandiri.
  - **Ketelitian:** Menekankan pentingnya sintaks yang benar dan logika yang presisi dalam pemrograman.
  - **Inovatif:** Memicu minat untuk menciptakan sesuatu yang baru atau meningkatkan yang sudah ada melalui kode.
  - **Komunikasi:** Melatih kemampuan menjelaskan logika program dan berkolaborasi dalam tim.

## **D DIMENSI PROFIL LULUSAN**

Berdasarkan tujuan pembelajaran, dimensi profil lulusan yang akan dicapai adalah:

- **Penalaran Kritis:** Peserta didik mampu merumuskan algoritma yang tepat untuk masalah tertentu, menganalisis kesalahan program, dan mengoptimalkan solusi.
- **Kreativitas:** Peserta didik mampu merancang program sederhana untuk memenuhi kebutuhan atau menyelesaikan masalah secara inovatif.
- **Kolaborasi:** Peserta didik mampu bekerja sama dengan teman dalam merancang algoritma, menulis kode, dan melakukan *debugging* bersama.
- **Kemandirian:** Peserta didik menunjukkan inisiatif dan ketekunan dalam belajar pemrograman, mencari solusi untuk *bug*, dan menyelesaikan proyek.
- **Komunikasi:** Peserta didik mampu menjelaskan logika algoritma dan kode program mereka secara jelas kepada orang lain.

## DESAIN PEMBELAJARAN

### A. CAPAIAN PEMBELAJARAN (CP) NOMOR : 32 TAHUN 2024

Pada akhir Fase E, peserta didik mampu menerapkan proses berpikir efektif dan efisien untuk menyelesaikan persoalan secara algoritmik sebagai solusi atas rancangan instruksi dan data yang dapat dijalankan secara efektif dan efisien oleh sistem komputasi, menerapkan berpikir kritis dalam menyikapi beragam data yang tersedia di internet untuk menjadi informasi yang bermanfaat, mempunyai wawasan tentang profesi informatika, serta memahami hak dan kewajiban sebagai warga digital dan aspek hukumnya. Capaian Pembelajaran setiap elemen adalah sebagai berikut.

| Elemen                 | Capaian Pembelajaran                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Berpikir Komputasional | <p>Peserta didik mampu memahami validitas sumber data; memahami konsep struktur data dan algoritma standar; menerapkan proses komputasi yang dilakukan manusia secara mandiri atau berkelompok untuk mendapatkan data yang bersih, benar, dan terpercaya; menerapkan struktur data dan algoritma standar untuk menghasilkan berbagai solusi dalam menyelesaikan persoalan yang mengandung himpunan data berstruktur kompleks dengan volume tidak kecil; serta menuliskan solusi rancangan program sederhana dalam format <i>pseudocode</i> yang dekat dengan bahasa komputer.</p> <p>Peserta didik mampu memahami model dan menyimulasikan dinamika Input-Proses-Output dalam sebuah komputer <i>Von Neumann</i>, serta memahami peran sistem operasi.</p>                                                                                                                                                                                                                                                                                                     |
| Literasi Digital       | <p>Peserta didik mampu memahami penggunaan mesin pencari dengan variabel yang lebih banyak; mengetahui ekosistem periksa fakta untuk memilah fakta dan bukan; menggunakan cara membaca lateral untuk mengevaluasi berbagai informasi digital; memahami pemanfaatan lebih beragam perangkat teknologi digital untuk membuat laporan, presentasi, serta analisis dan interpretasi data; memahami konsep dan penerapan serta konfigurasi keamanan dasar untuk konektivitas jaringan data lokal dan internet baik kabel maupun nirkabel; serta memahami pemanfaatan media digital untuk produksi dan diseminasi konten, partisipasi dan kolaborasi.</p> <p>Peserta didik mampu menghargai hak atas kekayaan intelektual, mengenal profesi bidang Informatika, memahami penerapan digitalisasi budaya Indonesia, menyaring konten negatif di dunia digital, menerapkan pengelolaan kata sandi dengan manajer kata sandi, dan menerapkan autentikasi dua langkah secara sederhana, serta menerapkan konfigurasi privasi dan keamanan pada akun platform digital.</p> |

### B. LINTAS DISIPLIN ILMU

- **Matematika:** Logika, aljabar, diskrit, fungsi, bilangan. Banyak masalah

pemrograman yang memerlukan pemahaman matematis.

- **Bahasa Inggris:** Banyak istilah dan sintaks dalam pemrograman berasal dari Bahasa Inggris.
- **Fisika/Kimia/Biologi:** Pemodelan simulasi, analisis data, visualisasi hasil eksperimen.
- **Desain Grafis/Seni:** Membuat aplikasi dengan antarmuka pengguna yang menarik, game, atau visualisasi.
- **Ekonomi/Bisnis:** Pembuatan aplikasi keuangan sederhana, analisis data penjualan, otomatisasi tugas.

## C. TUJUAN PEMBELAJARAN

Pertemuan 1: Pengantar Algoritma dan Pemrograman (Struktur Dasar)

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Menjelaskan perbedaan antara algoritma dan program komputer dengan tepat.
- Menyusun algoritma sederhana dalam bentuk pseudocode atau flowchart untuk masalah sehari-hari.
- Mengenal lingkungan pengembangan (IDE/editor online) untuk bahasa pemrograman yang digunakan (misal: Python).
- Menulis program "Hello World!" dan program sederhana untuk mencetak teks atau melakukan operasi aritmatika dasar.

Pertemuan 2: Variabel, Tipe Data, dan Operator

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Mendefinisikan variabel dan menjelaskan pentingnya dalam pemrograman.
- Mengidentifikasi berbagai tipe data (integer, float, string, boolean) dalam bahasa pemrograman yang digunakan.
- Menggunakan operator aritmatika, perbandingan, dan logika dalam ekspresi program.
- Menulis program sederhana yang melibatkan input dari pengguna, pemrosesan variabel, dan menampilkan output.

Pertemuan 3: Struktur Kontrol: Kondisional (If-Else)

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Menjelaskan konsep percabangan (kondisional) dalam algoritma dan program.
- Menggunakan pernyataan if, elif (atau else if), dan else untuk membuat program yang mengambil keputusan berdasarkan kondisi.
- Menyelesaikan masalah yang memerlukan pemilihan tindakan berdasarkan kriteria tertentu (misal: menentukan kelulusan, diskon harga).

Pertemuan 4: Struktur Kontrol: Perulangan (Looping)

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Menjelaskan konsep perulangan dalam algoritma dan program.
- Menggunakan pernyataan for dan while untuk melakukan iterasi dalam program.
- Menyelesaikan masalah yang memerlukan pengulangan tindakan sejumlah kali atau sampai kondisi tertentu terpenuhi (misal: menghitung jumlah deret, mencetak pola).

Pertemuan 5: Fungsi dan Modularitas Program

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Mendefinisikan fungsi sebagai blok kode yang dapat digunakan kembali.
- Membuat dan memanggil fungsi sederhana dalam program untuk memecah masalah menjadi bagian yang lebih kecil.
- Menjelaskan manfaat modularitas dalam pemrograman (keterbacaan, reusable, debugging).
- Melakukan *debugging* dasar untuk menemukan dan memperbaiki kesalahan dalam kode program sederhana.

#### Pertemuan 6: Proyek Pemrograman Sederhana dan Debugging

Setelah kegiatan pembelajaran, peserta didik diharapkan mampu:

- Menerapkan konsep algoritma dan pemrograman yang telah dipelajari untuk membuat proyek program sederhana (misal: kalkulator, game tebak angka, aplikasi *to-do list* sederhana).
- Menganalisis dan memperbaiki *bug* dalam program yang lebih kompleks secara mandiri atau kolaboratif.
- Mempresentasikan hasil proyek program dan menjelaskan logika di baliknya.

### D. TOPIK PEMBELAJARAN KONTEKSTUAL

- **Pencetakan Teks/Informasi:** Membuat program yang mencetak informasi pribadi, ucapan selamat, atau lirik lagu sederhana.
- **Kalkulator Sederhana:** Program untuk penjumlahan, pengurangan, perkalian, pembagian.
- **Konversi Satuan:** Program konversi suhu (Celcius ke Fahrenheit), panjang (meter ke kilometer).
- **Penentuan Kriteria:** Program untuk menentukan apakah seseorang lulus/tidak, diskon harga, status cuaca (hujan/tidak).
- **Perulangan/Pola:** Program untuk mencetak deret angka, pola bintang, atau menghitung akumulasi (misalnya, total belanja).
- **Game Sederhana:** Game tebak angka, permainan "batu-gunting-kertas" sederhana.
- **Aplikasi Mini:** Aplikasi untuk mengelola daftar belanja, daftar tugas sederhana, atau catatan digital.

### E. KERANGKA PEMBELAJARAN

#### 1. PRAKTIK PEDAGOGIK:

- **Model Pembelajaran:** *Discovery Learning* (untuk menemukan cara kerja sintaks/konsep), *Problem-Based Learning* (PBL) untuk menyelesaikan masalah dengan kode, dan *Project-Based Learning* (PjBL) untuk membuat aplikasi sederhana.
- **Strategi Pembelajaran:**
  - **Mindful Learning:** Fokus pada setiap baris kode, latihan *tracing* manual eksekusi kode, sesi *debugging* yang sistematis, refleksi terhadap *bug* sebagai peluang belajar.
  - **Meaningful Learning:** Mengaitkan setiap konsep pemrograman dengan contoh aplikasi nyata (misal: mengapa kita butuh perulangan di kalkulator?), studi kasus pembuatan program sederhana, dan mendorong peserta didik untuk membuat program yang relevan dengan minat mereka.

- **Joyful Learning:** Tantangan *coding* yang kompetitif (misal: siapa tercepat menyelesaikan *puzzle coding*), permainan *debugging* interaktif, membuat game sederhana, penggunaan platform visual yang menyenangkan, sesi *pair programming*.
- **Metode Pembelajaran:** Demonstrasi langsung (live coding), latihan terstruktur, *pair programming*, *code review*, diskusi kelompok, *debugging* bersama, proyek pemrograman.

## 2. KEMITRAAN PEMBELAJARAN:

- **Lingkungan Sekolah:** Guru TIK/komputer sebagai fasilitator, pengelola lab komputer. Anggota klub pemrograman atau ekstrakurikuler IT sebagai mentor sebaya.
- **Lingkungan Luar Sekolah:** Narasumber (programmer, *software developer*, desainer game) untuk berbagi pengalaman dan inspirasi (melalui daring atau kunjungan singkat). Komunitas *online coding* sebagai sumber belajar dan dukungan.
- **Masyarakat:** Mengidentifikasi masalah sehari-hari di lingkungan sekitar yang berpotensi diselesaikan dengan program sederhana.

## 3. LINGKUNGAN BELAJAR:

- **Ruang Fisik:** Laboratorium komputer dengan perangkat yang memadai dan koneksi internet stabil. Jika tidak ada, laptop/tablet pribadi peserta didik yang dapat diakses. Meja yang cukup luas untuk kerja kelompok. Proyektor untuk demonstrasi guru.
- **Ruang Virtual:** Platform Google Classroom untuk berbagi materi (slide, link IDE online, video tutorial), penugasan *coding*, dan pengumpulan proyek. IDE online (misal: Replit, Google Colab untuk Python) atau platform blok visual (Scratch, Blockly) sebagai lingkungan coding. Forum diskusi daring untuk tanya jawab dan *peer-support*.
- **Budaya Belajar:** Lingkungan yang mendukung eksperimen dan mencoba hal baru. Mendorong peserta didik untuk tidak takut membuat kesalahan (bug), melainkan melihatnya sebagai bagian alami dari proses belajar. Menekankan kolaborasi dan berbagi pengetahuan.

## 4. PEMANFAATAN DIGITAL:

- **Perpustakaan Digital:** Mengakses tutorial pemrograman online, dokumentasi bahasa pemrograman, e-book tentang algoritma dan pemrograman.
- **Forum Diskusi Daring:** Google Classroom atau platform lain untuk memposting *bug* yang ditemui, meminta bantuan, atau berbagi ide kode.
- **Penilaian Daring:** Kuis sintaks atau konsep dasar melalui Kahoot/Quizizz. Sistem *auto-grader* (jika tersedia) untuk tugas *coding* sederhana. Pengumpulan file program melalui Google Classroom.
- **Alat Interaktif:**
  - **IDE Online/Platform Coding:** Replit, Google Colab (Python), Scratch, Blockly (untuk pemula) sebagai lingkungan untuk menulis dan menjalankan kode.
  - **Situs Tutorial Interaktif:** Codecademy, freeCodeCamp, W3Schools untuk belajar sintaks dan konsep dasar secara interaktif.
  - **YouTube:** Menonton video tutorial pemrograman dan *live coding*.

## F. LANGKAH-LANGKAH PEMBELAJARAN BERDIFERENSIASI

### PERTEMUAN 1: PENGANTAR ALGORITMA DAN PEMROGRAMAN (STRUKTUR DASAR)

#### KEGIATAN PENDAHULUAN (15 MENIT)

- **Mindful Learning:** Guru memulai dengan pertanyaan: "Apa yang membuat sebuah aplikasi atau game bisa bekerja?" "Bagaimana komputer bisa mengerti perintah kita?" Guru meminta peserta didik untuk fokus pada ide "memberikan instruksi" kepada mesin.
- **Joyful Learning:** Guru menampilkan video singkat tentang sebuah robot yang melakukan tugas sederhana (misalnya, robot penyapu lantai) dan bertanya: "Menurut kalian, bagaimana cara robot ini tahu apa yang harus dilakukan?" Ini memicu rasa ingin tahu.
- **Meaningful Learning:** Guru mengaitkan ide instruksi ini dengan konsep "algoritma" dan "pemrograman" sebagai cara kita "berbicara" dengan komputer. Guru menyampaikan tujuan pembelajaran hari ini dan memperkenalkan bahasa pemrograman yang akan digunakan (misalnya Python sebagai bahasa teks pertama yang ramah pemula, atau Scratch/Blockly sebagai jembatan).

#### KEGIATAN INTI (60 MENIT)

- **Memahami (20 menit):**
  - Guru menjelaskan perbedaan antara algoritma (langkah-langkah umum) dan program (implementasi spesifik dalam bahasa komputer). Guru menampilkan contoh pseudocode dan flowchart sederhana untuk masalah "membuat kopi".
  - Guru mendemonstrasikan lingkungan pemrograman yang akan digunakan (IDE online) dan menunjukkan cara menulis serta menjalankan program "Hello World!".
- **Mengaplikasi (25 menit):**
  - Peserta didik secara individu atau berpasangan diminta untuk menulis program "Hello World!" pertama mereka di IDE online. (Diferensiasi proses: peserta didik yang lebih cepat dapat diminta untuk memodifikasi pesan atau menambahkan lebih banyak baris cetak).
  - Guru memberikan masalah sederhana (misalnya, "Cetak namamu 5 kali", "Cetak hasil penjumlahan  $2 + 3$ "). Peserta didik berlatih menulis program untuk masalah ini.
- **Merefleksi (15 menit):**
  - Guru meminta peserta didik menuliskan di kolom komentar Google Classroom: "Apa perasaanmu saat berhasil menjalankan program pertamamu?" dan "Apa tantangan terbesar saat menulis program 'Hello World!'?"
  - Beberapa peserta didik diminta untuk membagikan pengalaman mereka.

#### KEGIATAN PENUTUP (15 MENIT)

- **Umpan Balik Konstruktif:** Guru memberikan apresiasi atas keberanian peserta didik dalam menulis program pertama. Guru mengulas kembali konsep dasar dan potensi kesalahan umum (misalnya, *typo*).
- **Menyimpulkan Pembelajaran:** Peserta didik secara kolaboratif menyimpulkan bahwa pemrograman adalah cara kita memberikan instruksi kepada komputer,

dimulai dari algoritma yang jelas.

- **Perencanaan Pembelajaran Selanjutnya:** Guru memberikan pengantar untuk pertemuan berikutnya (variabel, tipe data, operator) dan memberikan tugas eksplorasi (mencari tahu berbagai jenis informasi yang bisa disimpan komputer).

## **PERTEMUAN 2: VARIABEL, TIPE DATA, DAN OPERATOR**

### **KEGIATAN PENDAHULUAN (15 MENIT)**

- **Mindful Learning:** Guru bertanya: "Bagaimana komputer bisa 'mengingat' nama kita atau angka yang kita masukkan?" Guru meminta peserta didik untuk fokus pada ide "menyimpan informasi".
- **Joyful Learning:** Guru dapat memainkan permainan "Simpan Informasi Rahasia" di mana setiap peserta didik memiliki "kotak" (variabel) untuk menyimpan informasi tertentu, dan mereka harus bertukar "kotak" sesuai aturan.
- **Meaningful Learning:** Guru mengaitkan aktivitas ini dengan konsep "variabel" dan "tipe data" dalam pemrograman, serta bagaimana komputer memanipulasi informasi tersebut dengan "operator". Guru menyampaikan tujuan pembelajaran.

### **KEGIATAN INTI (60 MENIT)**

- **Memahami (20 menit):**
  - Guru menjelaskan konsep variabel (sebagai wadah data) dan aturan penamaan variabel.
  - Guru memperkenalkan berbagai tipe data (integer, float, string, boolean) dengan contoh penggunaannya.
  - Guru menjelaskan berbagai jenis operator (aritmatika, perbandingan, logika) dan bagaimana mereka digunakan dalam ekspresi.
- **Mengaplikasi (25 menit):**
  - Peserta didik dalam kelompok kecil (diferensiasi proses: beberapa kelompok mendapatkan soal konversi tipe data yang lebih sederhana, yang lain dengan ekspresi logika yang lebih kompleks) mengerjakan latihan *coding* yang melibatkan:
    - Deklarasi dan inisialisasi variabel.
    - Melakukan operasi aritmatika dan mencetak hasilnya.
    - Mengambil input dari pengguna dan menyimpannya dalam variabel.
    - Membuat ekspresi perbandingan atau logika sederhana.
  - Guru berkeliling memberikan bantuan dan tips.
- **Merefleksi (15 menit):**
  - Guru meminta peserta didik menuliskan di jurnal: "Apa yang paling membingungkan tentang variabel dan tipe data?" dan "Bagaimana saya bisa menggunakan operator untuk membuat kalkulator sederhana?"

### **KEGIATAN PENUTUP (15 MENIT)**

- **Umpan Balik Konstruktif:** Guru memberikan umpan balik terhadap penggunaan variabel, tipe data, dan operator. Mengoreksi miskonsepsi umum.
- **Menyimpulkan Pembelajaran:** Guru dan peserta didik bersama-sama menyimpulkan pentingnya variabel untuk menyimpan data dan operator untuk

memanipulasi data tersebut.

- **Perencanaan Pembelajaran Selanjutnya:** Guru memberikan pengantar untuk pertemuan berikutnya (struktur kondisional) dan tugas latihan menulis program yang mengambil input dan melakukan perhitungan sederhana.

### **PERTEMUAN 3: STRUKTUR KONTROL: KONDISIONAL (IF-ELSE)**

#### **KEGIATAN PENDAHULUAN (15 MENIT)**

- **Mindful Learning:** Guru bertanya: "Bagaimana kita membuat keputusan dalam hidup? (misalnya, jika hujan, maka bawa payung). Bagaimana komputer bisa membuat keputusan serupa?" Guru meminta peserta didik untuk fokus pada ide "membuat pilihan".
- **Joyful Learning:** Guru dapat memulai dengan permainan "Siapa yang Berhak Lewat?" di mana peserta didik harus mengikuti aturan if-else (misalnya, jika siswa mengenakan baju biru, dia boleh lewat; jika tidak, dia harus duduk).
- **Meaningful Learning:** Guru mengaitkan aktivitas ini dengan konsep "kondisional" atau "percabangan" dalam pemrograman. Guru menyampaikan tujuan pembelajaran.

#### **KEGIATAN INTI (60 MENIT)**

- **Memahami (20 menit):**
  - Guru menjelaskan konsep struktur kondisional (if, elif/else if, else) dengan contoh diagram alir dan kode.
  - Guru mendemonstrasikan bagaimana program dapat menjalankan blok kode yang berbeda berdasarkan suatu kondisi.
- **Mengaplikasi (25 menit):**
  - Peserta didik dalam kelompok (diferensiasi konten: beberapa kelompok mengerjakan soal penentuan diskon sederhana, yang lain dengan kondisi bersarang/lebih kompleks) mengerjakan latihan *coding* yang melibatkan:
    - Program penentu kelulusan berdasarkan nilai.
    - Program penentu diskon berdasarkan total belanja.
    - Program penentu jenis bilangan (positif, negatif, nol).
  - Mereka didorong untuk menguji berbagai kasus input untuk memastikan program berjalan benar.
- **Merefleksi (15 menit):**
  - Guru meminta peserta didik merefleksikan: "Apa yang paling menantang dari menulis kondisi dalam program?" dan "Berikan satu contoh masalah sehari-hari yang akan lebih mudah dipecahkan dengan pernyataan if-else."

#### **KEGIATAN PENUTUP (15 MENIT)**

- **Umpan Balik Konstruktif:** Guru memberikan umpan balik terhadap logika kondisional peserta didik. Mengoreksi kesalahan umum dalam penulisan kondisi atau indentasi (jika menggunakan Python).
- **Menyimpulkan Pembelajaran:** Guru dan peserta didik bersama-sama menyimpulkan bahwa kondisional memungkinkan program untuk membuat keputusan.
- **Perencanaan Pembelajaran Selanjutnya:** Guru memberikan pengantar untuk

pertemuan berikutnya (struktur perulangan) dan tugas latihan membuat program yang menerapkan kondisional untuk masalah sederhana (misalnya, detektor angka ganjil/genap).

#### **PERTEMUAN 4: STRUKTUR KONTROL: PERULANGAN (LOOPING)**

##### **KEGIATAN PENDAHULUAN (15 MENIT)**

- **Mindful Learning:** Guru bertanya: "Bayangkan jika kita harus mencetak 'Hello!' 1000 kali. Apakah kita akan menuliskannya 1000 kali? Bagaimana cara yang lebih efisien?" Guru meminta peserta didik untuk fokus pada ide "pengulangan efisien".
- **Joyful Learning:** Guru dapat memulai dengan demonstrasi fisik sederhana yang berulang (misalnya, melompat 5 kali, tepuk tangan 3 kali) dan bertanya: "Bagaimana saya bisa memberitahu komputer untuk melakukan ini?"
- **Meaningful Learning:** Guru mengaitkan aktivitas ini dengan konsep "perulangan" atau "looping" dalam pemrograman. Guru menyampaikan tujuan pembelajaran.

##### **KEGIATAN INTI (60 MENIT)**

- **Memahami (20 menit):**
  - Guru menjelaskan konsep perulangan (for loop dan while loop) dengan contoh diagram alir dan kode. Guru menjelaskan kapan menggunakan for (jumlah iterasi diketahui) dan while (jumlah iterasi tidak diketahui/berdasarkan kondisi).
  - Guru mendemonstrasikan bagaimana perulangan dapat mengotomatisasi tugas yang berulang.
- **Mengaplikasi (25 menit):**
  - Peserta didik dalam kelompok (diferensiasi konten: beberapa kelompok mengerjakan soal deret sederhana, yang lain dengan pola perulangan bertingkat atau penggunaan break/continue) mengerjakan latihan *coding* yang melibatkan:
    - Mencetak angka dari 1 sampai 10.
    - Menghitung jumlah deret.
    - Mencetak pola bintang atau angka.
    - Program tebak angka sederhana (menggunakan while loop).
  - Mereka didorong untuk memvisualisasikan bagaimana *loop* bekerja.
- **Merefleksi (15 menit):**
  - Guru meminta peserta didik merefleksikan: "Kapan saya harus menggunakan for loop dan kapan while loop?" dan "Apa manfaat terbesar dari perulangan dalam pemrograman?"

##### **KEGIATAN PENUTUP (15 MENIT)**

- **Umpan Balik Konstruktif:** Guru memberikan umpan balik terhadap penggunaan perulangan. Mengoreksi kesalahan umum seperti *infinite loop* atau kondisi terminasi yang salah.
- **Menyimpulkan Pembelajaran:** Guru dan peserta didik bersama-sama menyimpulkan bahwa perulangan adalah alat yang ampuh untuk mengotomatisasi tugas berulang.
- **Perencanaan Pembelajaran Selanjutnya:** Guru memberikan pengantar untuk pertemuan berikutnya (fungsi dan modularitas) dan tugas latihan membuat program

yang menggunakan perulangan.

## **PERTEMUAN 5: FUNGSI DAN MODULARITAS PROGRAM**

### **KEGIATAN PENDAHULUAN (15 MENIT)**

- **Mindful Learning:** Guru bertanya: "Ketika kita memasak, kita sering menggunakan resep yang sama untuk bagian tertentu (misalnya, membuat saus). Bagaimana kita bisa menerapkan ide ini dalam pemrograman agar kode kita rapi dan mudah digunakan kembali?" Guru meminta peserta didik untuk fokus pada ide "mengorganisir kode".
- **Joyful Learning:** Guru dapat mempresentasikan sebuah program yang sangat panjang dan sulit dibaca, lalu menampilkan versi yang sama tetapi dengan fungsi yang terstruktur, dan bertanya: "Mana yang lebih mudah dipahami?"
- **Meaningful Learning:** Guru mengaitkan ide "reusable code" ini dengan konsep "fungsi" dan "modularitas" dalam pemrograman. Guru menyampaikan tujuan pembelajaran.

### **KEGIATAN INTI (60 MENIT)**

- **Memahami (20 menit):**
  - Guru menjelaskan konsep fungsi (mendefinisikan dan memanggil). Guru menjelaskan manfaat modularitas (kode lebih rapi, mudah dibaca, mudah di-debug, *reusable*).
  - Guru mendemonstrasikan cara membuat fungsi sederhana (misalnya, fungsi untuk menjumlahkan dua angka, fungsi untuk mencetak pesan sapaan).
  - Guru memperkenalkan teknik *debugging* dasar (misal: menggunakan print statement, membaca pesan error).
- **Mengaplikasi (25 menit):**
  - Peserta didik dalam kelompok (diferensiasi proses: beberapa kelompok membuat fungsi sederhana tanpa parameter, yang lain dengan parameter dan nilai kembalian, atau bahkan fungsi bersarang) mengerjakan latihan *coding* yang melibatkan:
    - Membuat program yang menggunakan fungsi untuk setiap operasi (input, proses, output).
    - Membagi program sebelumnya menjadi beberapa fungsi.
    - Melakukan *debugging* pada kode yang sengaja disisipi *bug* sederhana oleh guru.
  - Mereka didorong untuk berdiskusi dan membantu satu sama lain dalam *debugging*.
- **Merefleksi (15 menit):**
  - Guru meminta peserta didik merefleksikan: "Bagaimana fungsi membuat program saya lebih mudah dipahami?" dan "Apa tips terbaik saya untuk menemukan *bug*?"

### **KEGIATAN PENUTUP (15 MENIT)**

- **Umpan Balik Konstruktif:** Guru memberikan umpan balik terhadap penggunaan fungsi dan kemampuan *debugging*.

- **Menyimpulkan Pembelajaran:** Guru dan peserta didik bersama-sama menyimpulkan bahwa fungsi dan modularitas sangat penting untuk membuat program yang baik dan mudah dikelola.
- **Perencanaan Pembelajaran Selanjutnya:** Guru memberitahu bahwa pertemuan terakhir adalah proyek akhir dan persiapan presentasi.

## **PERTEMUAN 6: PROYEK PEMROGRAMAN SEDERHANA DAN DEBUGGING**

### **KEGIATAN PENDAHULUAN (15 MENIT)**

- **Mindful Learning:** Guru bertanya: "Setelah belajar semua konsep ini, aplikasi apa yang paling ingin kalian buat?" Guru meminta peserta didik untuk fokus pada ide "menciptakan sesuatu".
- **Joyful Learning:** Guru menampilkan beberapa contoh proyek sederhana yang dapat dibuat oleh pemula (misalnya, kalkulator BMI, game tebak angka dengan *feedback*, program penghitung rata-rata nilai).
- **Meaningful Learning:** Guru menekankan bahwa tujuan akhir dari belajar pemrograman adalah untuk menciptakan solusi. Guru menyampaikan instruksi proyek dan rubrik penilaian.

### **KEGIATAN INTI (60 MENIT)**

- **Memahami (20 menit):**
  - Guru menjelaskan secara rinci tugas proyek: setiap kelompok (atau individu, tergantung kebutuhan) memilih satu masalah sederhana dan merancang programnya menggunakan konsep yang telah dipelajari (variabel, kondisional, perulangan, fungsi).
  - Mereka harus membuat perencanaan (algoritma/flowchart) dan kemudian menulis kode programnya.
  - Guru memberikan daftar masalah yang dapat dipilih atau siswa dapat mengusulkan masalah sendiri.
- **Mengaplikasi (25 menit):**
  - Peserta didik bekerja secara kolaboratif (jika berkelompok) atau mandiri untuk mengembangkan proyek mereka. Guru berperan sebagai fasilitator dan *debugger* ahli, memberikan bimbingan sesuai kebutuhan (diferensiasi dukungan).
  - Mereka harus menguji program mereka secara menyeluruh dan melakukan *debugging* jika ada kesalahan.
- **Merefleksi (15 menit):**
  - Setiap kelompok/individu mempresentasikan proyek mereka, mendemonstrasikan programnya, dan menjelaskan logika di baliknya.
  - Guru memfasilitasi diskusi refleksi akhir: "Apa tantangan terbesar saat membuat proyek ini dan bagaimana kalian mengatasinya?" "Apa yang paling kalian banggakan dari proyek ini?"

### **KEGIATAN PENUTUP (15 MENIT)**

- **Umpan Balik Konstruktif:** Guru memberikan umpan balik individu dan kelompok terhadap proyek. Guru menyoroti kekuatan program (misalnya, logika yang efisien,

UI yang baik, kode yang bersih) dan area yang masih bisa ditingkatkan.

- **Menyimpulkan Pembelajaran:** Guru dan peserta didik bersama-sama menyimpulkan bahwa pemrograman adalah keterampilan yang kuat untuk memecahkan masalah dan menciptakan inovasi.
- **Perencanaan Pembelajaran Selanjutnya:** Guru mendorong peserta didik untuk terus berlatih dan menjelajahi dunia pemrograman lebih jauh (misalnya, mengikuti *bootcamp* online, bergabung dengan komunitas *coding*).

## G. ASESMEN PEMBELAJARAN

Asesmen akan dilakukan secara komprehensif untuk mengukur pencapaian kompetensi peserta didik.

### ASSESSMENT AS LEARNING (AS):

- **Self-Assessment:** Jurnal refleksi setelah setiap sesi *coding* tentang pemahaman konsep, *bug* yang ditemui dan cara mengatasinya, serta strategi belajar pemrograman yang efektif.
- **Peer Assessment:** Peserta didik saling melakukan *code review* terhadap program teman, memberikan masukan konstruktif, dan menilai partisipasi dalam kelompok.
- **Diskusi Kode:** Observasi partisipasi aktif dalam diskusi tentang logika kode, *debugging*, dan berbagi solusi.

### ASSESSMENT FOR LEARNING (AFL):

- **Kuis Singkat/Tanya Jawab Lisan:** Menguji pemahaman sintaks dasar, definisi variabel, fungsi operator, atau identifikasi tipe data (menggunakan Kahoot/Quizizz).
- **Latihan Coding Harian/Mingguan:** Memberikan tugas *coding* singkat sebagai latihan untuk mengidentifikasi pemahaman dan kesulitan siswa secara formatif, serta memberikan umpan balik langsung.
- **Umpan Balik Guru:** Memberikan umpan balik langsung saat *live coding*, saat *debugging* bersama, atau saat mengoreksi program.
- **Observasi:** Guru mengamati kemampuan peserta didik dalam merancang algoritma, menulis kode, mengidentifikasi *bug*, dan menerapkan konsep pemrograman.

### ASSESSMENT OF LEARNING (AOL):

- **Tes Tertulis:** Soal-soal yang mencakup:
  - Menulis pseudocode atau flowchart untuk masalah tertentu.
  - Menganalisis potongan kode dan memprediksi outputnya.
  - Mengidentifikasi kesalahan (*bug*) dalam kode program.
  - Menjelaskan konsep pemrograman (variabel, tipe data, kondisional, perulangan, fungsi).
- **Penilaian Kinerja/Proyek Pemrograman:**
  - **Proyek Akhir:** Penilaian terhadap program yang dihasilkan (berfungsi dengan benar, memenuhi spesifikasi, kode bersih/terbaca, efisien) dan presentasinya (kemampuan menjelaskan logika, proses *debugging*, dan interpretasi hasil). Rubrik penilaian proyek mencakup aspek ketepatan, fungsionalitas, keterbacaan kode, dan kreativitas (sesuai dengan kriteria di buku ajar pada halaman 128).
  - **Tantangan Debugging:** Peserta didik diberikan program dengan *bug* dan diminta untuk menemukan serta memperbaikinya dalam waktu tertentu. Penilaian didasarkan pada ketepatan dan efisiensi perbaikan.

- **Portofolio Digital:** Kumpulan semua program yang telah dibuat, laporan proyek, dan jurnal refleksi yang menunjukkan perkembangan kompetensi peserta didik.