Pengenalan Conditional Statements

Slide 1

- **Apa itu Conditional Statements?**
  - Pernyataan yang memungkinkan program membuat keputusan
  - Program dapat menjalankan kode yang berbeda berdasarkan kondisi tertentu
  - Membuat program menjadi interaktif dan dinamis
- **Mengapa Conditional Statements Penting?**
  - **Real-world decision making**: Seperti manusia yang membuat keputusan setiap hari
  - **Program logic**: Menentukan alur jalannya program
  - **User interaction**: Merespons input user dengan tepat
  - **Data validation**: Memeriksa apakah data valid atau tidak
- **Contoh Aplikasi Sehari-hari:**

```
// Contoh: Sistem login sederhana
if (password == "12345") {
    cout << "Login berhasil!" << endl;
} else {
    cout << "Password salah!" << endl;
}

// Contoh: Cek cuaca untuk rekomendasi pakaian
if (suhu < 20) {
    cout << "Pakai jaket tebal" << endl;
} else if (suhu < 30) {
    cout << "Pakai kaos lengan panjang" << endl;
} else {
    cout << "Pakai kaos" << endl;
}
```

- **Jenis-jenis Conditional Statements di C++:**
    - `if` statement
    - `if-else` statement
    - `else-if` (elif) statement
    - `nested if` statement
    - `switch` statement (akan dipelajari terpisah)

If Statement - Dasar Conditional

Slide 2

- **Sintaks If Statement:**

```
if (kondisi) {
    // blok kode yang dijalankan jika kondisi true
}
```

- **Komponen If Statement:**
  - `if` : Kata kunci untuk memulai conditional
  - `(kondisi)` : Ekspresi boolean yang dievaluasi (true/false)
  - `{}` : Kurung kurawal untuk mengelompokkan kode
  - **Blok kode**: Instruksi yang dijalankan jika kondisi true
- **Contoh Sederhana:**

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

int main() {
    int nilai = 85;

    if (nilai >= 75) {
        cout << "Selamat! Anda lulus!" << endl;
    }

    cout << "Program selesai" << endl;
    return 0;
}
```

Output:

```
Selamat! Anda lulus!
Program selesai
```

- **Contoh dengan Input User:**

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

```
int main() {
    int umur;

    cout << "Masukkan umur Anda: ";
    cin >> umur;

    if (umur >= 17) {
        cout << "Anda sudah bisa punya KTP!" << endl;
    }

    if (umur >= 18) {
        cout << "Anda sudah bisa memilih dalam Pemilu!" << endl;
    }

    return 0;
}
```

- **Penting Diingat:**
  - Jika kondisi `false`, blok kode dalam if diabaikan
  - Program tetap melanjutkan ke baris berikutnya setelah if
  - Kondisi harus menghasilkan nilai boolean (true/false)

Else Statement - Alternatif Pilihan

Slide 3

- **Sintaks If-Else Statement:**

```
if (kondisi) {
    // blok kode jika kondisi true
} else {
    // blok kode jika kondisi false
}
```

- **Kapan Gunakan Else?**
  - Ketika ada **2 pilihan** yang saling eksklusif
  - Salah satu blok kode **pasti dijalankan**
  - Memberikan alternatif jika kondisi if tidak terpenuhi
- **Contoh Klasik - Ganjil/Genap:**

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

int main() {
    int angka;

    cout << "Masukkan sebuah angka: ";
    cin >> angka;

    if (angka % 2 == 0) {
        cout << angka << " adalah bilangan genap" << endl;
    } else {
        cout << angka << " adalah bilangan ganjil" << endl;
    }

    return 0;
}
```

Contoh Output:

```
Masukkan sebuah angka: 7
7 adalah bilangan ganjil

Masukkan sebuah angka: 12
12 adalah bilangan genap
```

- **Contoh Praktis - Sistem Penilaian:**

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

int main() {
    double nilai;

    cout << "Masukkan nilai ujian: ";
    cin >> nilai;

    if (nilai >= 60) {
        cout << "LULUS - Selamat!" << endl;
        cout << "Nilai Anda: " << nilai << endl;
    } else {
        cout << "TIDAK LULUS - Belajar lebih giat!" << endl;
        cout << "Nilai Anda: " << nilai << endl;
        cout << "Nilai minimum untuk lulus: 60" << endl;
    }

    return 0;
}
```

- **Keuntungan If-Else:**
  - **Komprehensif**: Semua kemungkinan tercakup

- **Jelas**: Logic yang mudah dibaca dan dipahami
- **Efisien**: Hanya satu kondisi yang dievaluasi

Else-If Statement - Multiple Conditions

Slide 4

- **Sintaks Else-If Statement:**

```
if (kondisi1) {
    // blok kode untuk kondisi1
} else if (kondisi2) {
    // blok kode untuk kondisi2
} else if (kondisi3) {
    // blok kode untuk kondisi3
} else {
    // blok kode default (opsional)
}
```

- **Karakteristik Else-If:**
  - Memungkinkan **multiple conditions** dalam satu struktur
  - Dievaluasi **secara berurutan** dari atas ke bawah
  - **Hanya satu blok** yang dijalankan (yang pertama bernilai true)

- Evaluasi **berhenti** setelah menemukan kondisi true

- **Contoh: Sistem Grading**

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

int main() {
    double nilai;

    cout << "Masukkan nilai (0-100): ";
    cin >> nilai;

    if (nilai >= 90) {
        cout << "Grade: A (Excellent!)" << endl;
    } else if (nilai >= 80) {
        cout << "Grade: B (Good)" << endl;
    } else if (nilai >= 70) {
        cout << "Grade: C (Average)" << endl;
    } else if (nilai >= 60) {
        cout << "Grade: D (Below Average)" << endl;
    } else {
        cout << "Grade: F (Fail)" << endl;
    }

    return 0;
```

```
}
```

- **Contoh: Kategori Umur**

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

int main() {
    int umur;

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

    if (umur < 0) {
        cout << "Umur tidak valid!" << endl;
    } else if (umur <= 5) {
        cout << "Kategori: Balita" << endl;
    } else if (umur <= 12) {
        cout << "Kategori: Anak-anak" << endl;
    } else if (umur <= 17) {
        cout << "Kategori: Remaja" << endl;
    } else if (umur <= 60) {
        cout << "Kategori: Dewasa" << endl;
    } else {
        cout << "Kategori: Lansia" << endl;
    }
```

```
    return 0;
}
```

Contoh Output:

```
Masukkan umur: 15
Kategori: Remaja

Masukkan umur: 25
Kategori: Dewasa
```

- **Tips Penting:**
  - **Urutan kondisi** sangat penting (dari spesifik ke umum)
  - Gunakan `else` terakhir sebagai **default case**
  - Pastikan kondisi **tidak overlap** secara tidak sengaja

Nested If Statements - If di Dalam If

Slide 5

- **Apa itu Nested If?**
  - If statement yang ditulis **di dalam** if statement lain
  - Memungkinkan pemeriksaan kondisi **bertingkat**
  - Berguna untuk logic yang **kompleks** dan **hierarkis**
- **Sintaks Nested If:**

```
if (kondisi_luar) {
    // kode untuk kondisi luar

    if (kondisi_dalam) {
        // kode untuk kondisi dalam
    } else {
        // kode alternatif kondisi dalam
    }

} else {
```

```
    // kode alternatif kondisi luar
}
```

- **Contoh: Sistem Beasiswa**

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

int main() {
    double ipk;
    int semester;
    bool aktiveMahasiswa;

    cout << "Masukkan IPK: ";
    cin >> ipk;
    cout << "Masukkan semester: ";
    cin >> semester;
    cout << "Aktif sebagai mahasiswa? (1=Ya, 0=Tidak): ";
    cin >> aktiveMahasiswa;

    if (aktiveMahasiswa) {
        cout << "Status: Mahasiswa aktif" << endl;

        if (ipk >= 3.5) {
            cout << "IPK memenuhi syarat" << endl;

            if (semester >= 3) {
```

```
                cout << "SELAMAT! Anda berhak mendapat beasiswa!" << endl;
                cout << "Jenis beasiswa: Prestasi Akademik" << endl;
            } else {
                cout << "Maaf, semester belum mencukupi (min. semester 3)" << endl;
            }

        } else {
            cout << "Maaf, IPK belum mencukupi (min. 3.5)" << endl;
        }

    } else {
        cout << "Maaf, hanya untuk mahasiswa aktif" << endl;
    }

    return 0;
}
```

- **Contoh: Login Multi-Level**

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

int main() {
    string username, password;
    int level;
```

```
    cout << "Username: ";
    cin >> username;
    cout << "Password: ";
    cin >> password;

    if (username == "admin" && password == "admin123") {
        cout << "Login berhasil sebagai Admin" << endl;

        cout << "Pilih level akses (1=Read, 2=Write, 3=Delete): ";
        cin >> level;

        if (level == 1) {
            cout << "Akses: Read Only" << endl;
        } else if (level == 2) {
            cout << "Akses: Read & Write" << endl;
        } else if (level == 3) {
            cout << "Akses: Full Control (Read, Write, Delete)" << endl;
        } else {
            cout << "Level tidak valid" << endl;
        }

    } else if (username == "user" && password == "user123") {
        cout << "Login berhasil sebagai User" << endl;
        cout << "Akses: Read Only" << endl;

    } else {
        cout << "Username atau password salah!" << endl;
    }
```

```
    return 0;
}
```

Tips Nested If:

- **Indentasi** yang konsisten untuk readability
- Hindari nesting **terlalu dalam** (max 3-4 level)
- Pertimbangkan alternatif seperti **logical operators**

Kombinasi Complex Conditionals

Slide 6

- **Menggabungkan Multiple Approaches:**
  - Nested if + else-if dalam satu program
  - Multiple conditions dengan logical operators
  - Real-world scenarios yang kompleks
- **Contoh: Sistem Penerimaan Mahasiswa**

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

int main() {
    double nilaiMatematika, nilaiBahasaInggris, nilaiIPA;
    string jurusan;
    bool lulusWawancara;

    cout << "=== SISTEM PENERIMAAN MAHASISWA ===" << endl;
    cout << "Masukkan nilai Matematika (0-100): ";
```

```
    cin >> nilaiMatematika;
    cout << "Masukkan nilai Bahasa Inggris (0-100): ";
    cin >> nilaiBahasaInggris;
    cout << "Masukkan nilai IPA (0-100): ";
    cin >> nilaiIPA;
    cout << "Masukkan jurusan yang diinginkan (TEKNIK/EKONOMI/SASTRA): ";
    cin >> jurusan;
    cout << "Lulus wawancara? (1=Ya, 0=Tidak): ";
    cin >> lulusWawancara;

    // Hitung rata-rata
    double rataRata = (nilaiMatematika + nilaiBahasaInggris + nilaiIPA) / 3.0;

    cout << "\n=== HASIL EVALUASI ===" << endl;
    cout << "Rata-rata nilai: " << rataRata << endl;

    if (rataRata >= 70.0) {
        cout << "? Syarat nilai akademik terpenuhi" << endl;

        if (lulusWawancara) {
            cout << "? Lulus wawancara" << endl;

            // Cek syarat khusus per jurusan
            if (jurusan == "TEKNIK") {
                if (nilaiMatematika >= 80 && nilaiIPA >= 75) {
                    cout << "SELAMAT! Anda DITERIMA di jurusan TEKNIK" << endl;
                } else {
                    cout << "Maaf, nilai Matematika (min.80) atau IPA (min.75) kurang" << endl;
                }
```

```
            } else if (jurusan == "EKONOMI") {
                if (nilaiMatematika >= 75 && nilaiBahasaInggris >= 70) {
                    cout << "SELAMAT! Anda DITERIMA di jurusan EKONOMI" << endl;
                } else {
                    cout << "Maaf, nilai Matematika (min.75) atau B.Inggris (min.70) kurang" << endl;
                }
            } else if (jurusan == "SASTRA") {
                if (nilaiBahasaInggris >= 85) {
                    cout << "SELAMAT! Anda DITERIMA di jurusan SASTRA" << endl;
                } else {
                    cout << "Maaf, nilai Bahasa Inggris harus minimal 85" << endl;
                }
            } else {
                cout << "Jurusan tidak tersedia" << endl;
            }

        } else {
            cout << "? Tidak lulus wawancara" << endl;
            cout << "Maaf, Anda tidak diterima" << endl;
        }

    } else {
        cout << "? Nilai akademik tidak memenuhi syarat (min. 70)" << endl;
        cout << "Maaf, Anda tidak diterima" << endl;
    }

    return 0;
}
```

Contoh Output:

```
=== SISTEM PENERIMAAN MAHASISWA ===
Masukkan nilai Matematika (0-100): 85
Masukkan nilai Bahasa Inggris (0-100): 78
Masukkan nilai IPA (0-100): 82
Masukkan jurusan yang diinginkan (TEKNIK/EKONOMI/SASTRA): TEKNIK
Lulus wawancara? (1=Ya, 0=Tidak): 1

=== HASIL EVALUASI ===
Rata-rata nilai: 81.67
? Syarat nilai akademik terpenuhi
? Lulus wawancara
SELAMAT! Anda DITERIMA di jurusan TEKNIK
```

- **Key Learning Points:**
  - **Hierarkis**: Cek syarat umum dulu, baru syarat spesifik
  - **Modular**: Pisahkan logic menjadi bagian-bagian yang jelas
  - **User-friendly**: Berikan feedback yang informatif

Operator Prioritas & Precedence

Slide 7

- **Operator Prioritas dalam C++:**

Prioritas	Operator	Deskripsi	Contoh
1 (Highest)	`()`	Parentheses	`(a + b)`
2	`!`	Logical NOT	`!isValid`
3	`*`, `/`, `%`	Multiply, Divide, Modulo	`a * b / c`
4	`+`, `-`	Addition, Subtraction	`a + b - c`
5	`<`, `<=`, `>`, `>=`	Relational	`a < b`
6	`==`, `!=`	Equality	`a == b`
7	`&&`	Logical AND	`a && b`

Prioritas	Operator	Deskripsi	Contoh
8 (Lowest)	`||`	Logical OR	`a || b`

- **Contoh Tanpa Parentheses:**

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

int main() {
    int a = 10, b = 5, c = 3;

    // Tanpa parentheses - ikuti operator precedence
    if (a + b > c * 5 && b != 0) {
        //   15   > 15     && true
        //   false        && true  = false
        cout << "Kondisi 1: FALSE" << endl;
    }

    if (a > b && b > c || a == 10) {
        // true && true || true
        //      true     || true  = true
        cout << "Kondisi 2: TRUE" << endl;
    }
```

```
    if (!a > b || b * c < a) {
        // !10 > 5 || 15 < 10
        //  0  > 5 || false
        // false   || false = false
        cout << "Kondisi 3: FALSE" << endl;
    }

    return 0;
}
```

- **Contoh Dengan Parentheses untuk Clarity:**

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

int main() {
    int nilai = 85;
    int kehadiran = 90;
    bool tugasLengkap = true;

    // Tanpa parentheses (sesuai precedence)
    if (nilai >= 80 && kehadiran >= 85 || tugasLengkap) {
        cout << "Versi 1: Lulus" << endl;
    }

    // Dengan parentheses untuk clarity
    if ((nilai >= 80) && (kehadiran >= 85) || tugasLengkap) {
```

```
        cout << "Versi 2: Lulus (sama dengan versi 1)" << endl;
    }

    // Mengubah makna dengan parentheses
    if (nilai >= 80 && (kehadiran >= 85 || tugasLengkap)) {
        cout << "Versi 3: Lulus (makna berbeda)" << endl;
    }

    return 0;
}
```

- **Best Practice:**

```
// ? Sulit dibaca
if (a > b && c < d || e == f && !g)

// ? Mudah dibaca
if ((a > b && c < d) || (e == f && !g))

// ? Lebih mudah dibaca lagi
if ((a > b && c < d) ||
    (e == f && !g))
```

- **Tips Penting:**

- **Gunakan parentheses** untuk clarity, meskipun tidak wajib
- **AND (`&&`) lebih prioritas** daripada OR (`||`)
- **Relational operators** (`<`, `>`, `==`) lebih prioritas dari logical
- Jika ragu, **tambahkan parentheses**!

Logical Operators & Short-Circuit Evaluation

Slide 8

- **Logical Operators dalam C++:**

Operator	Nama	Deskripsi	Contoh
`&&`	AND	True jika KEDUA kondisi true	`a > 5 && b < 10`
`||`	OR	True jika SALAH SATU kondisi true	`a == 0 || b == 0`
`!`	NOT	Membalik nilai boolean	`!(a > b)`

- **Truth Table:**

```
A     | B     | A && B | A || B | !A
------|-------|--------|--------|----
true  | true  | true   | true   | false
true  | false | false  | true   | false
false | true  | false  | true   | true
false | false | false  | false  | true
```

- **Short-Circuit Evaluation:**

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

int main() {
    int a = 0, b = 5;

    // Short-circuit dengan &&
    // Jika kondisi pertama false, kondisi kedua TIDAK dievaluasi
    if (a != 0 && (b / a) > 2) {
        cout << "Ini tidak akan dijalankan" << endl;
    } else {
        cout << "Short-circuit menyelamatkan dari division by zero!" << endl;
    }

    // Short-circuit dengan ||
    // Jika kondisi pertama true, kondisi kedua TIDAK dievaluasi
    if (a == 0 || (b / a) > 2) {
        cout << "Kondisi pertama true, tidak perlu cek kedua" << endl;
    }

    return 0;
}
```

- **Contoh Praktis: Validasi Input**

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

int main() {
    int umur, gaji;
    bool penyakitKronis;

    cout << "Masukkan umur: ";
    cin >> umur;
    cout << "Masukkan gaji (juta): ";
    cin >> gaji;
    cout << "Punya penyakit kronis? (1=Ya, 0=Tidak): ";
    cin >> penyakitKronis;

    // Cek kelayakan asuransi
    if (umur >= 18 && umur <= 60 &&
        gaji >= 3 &&
        !penyakitKronis) {

        cout << "? Anda layak mendapat asuransi" << endl;

        // Tentukan premi berdasarkan kondisi
        if (umur < 30 && gaji >= 10) {
            cout << "Premi: Diskon 20% (Usia muda + Gaji tinggi)" << endl;
        } else if (umur < 30 || gaji >= 10) {
            cout << "Premi: Diskon 10% (Salah satu syarat terpenuhi)" << endl;
        } else {
            cout << "Premi: Normal" << endl;
```

```
        }

    } else {
        cout << "? Maaf, Anda belum memenuhi syarat" << endl;

        // Berikan feedback spesifik
        if (umur < 18) {
            cout << "Alasan: Umur terlalu muda (min. 18)" << endl;
        } else if (umur > 60) {
            cout << "Alasan: Umur terlalu tua (max. 60)" << endl;
        }

        if (gaji < 3) {
            cout << "Alasan: Gaji tidak mencukupi (min. 3 juta)" << endl;
        }

        if (penyakitKronis) {
            cout << "Alasan: Memiliki penyakit kronis" << endl;
        }
    }

    return 0;
}
```

- **Complex Logical Expressions:**

```
// Contoh: Sistem booking hotel
bool weekday = true;
```

```
bool member = false;
int hari = 3; // 1-7 (1=Senin)
double budget = 500000;

if ((weekday && budget >= 300000) ||
    (!weekday && budget >= 500000) ||
    (member && budget >= 250000)) {
    cout << "Kamar tersedia sesuai budget" << endl;
}
```

Edge Cases & Common Pitfalls

Slide 9

- **Edge Cases yang Sering Diabaikan:**

1. Division by Zero

```
// ? Bahaya - bisa crash program
int a = 10, b = 0;
if (a / b > 5) {  // ERROR: Division by zero!
    cout << "Kondisi terpenuhi" << endl;
}

// ? Aman - cek dulu sebelum bagi
if (b != 0 && a / b > 5) {
    cout << "Kondisi terpenuhi" << endl;
}
```

2. Floating Point Comparison

```
// ? Tidak reliable karena floating point precision
double nilai = 0.1 + 0.2;  // Hasil: 0.30000000000000004
if (nilai == 0.3) {
```

```
    cout << "Sama" << endl;  // Mungkin tidak ter-print
}

// ? Gunakan tolerance untuk floating point
double tolerance = 0.000001;
if (abs(nilai - 0.3) < tolerance) {
    cout << "Sama (dengan tolerance)" << endl;
}
```

3. String Comparison Case Sensitivity

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

int main() {
    string input;
    cout << "Masukkan jawaban (ya/tidak): ";
    cin >> input;

    // ? Case sensitive - "Ya", "YA", "yA" tidak akan match
    if (input == "ya") {
        cout << "Anda menjawab ya" << endl;
    }

    // ? Convert ke lowercase dulu
```

```
    transform(input.begin(), input.end(), input.begin(), ::tolower);
    if (input == "ya" || input == "y") {
        cout << "Anda menjawab ya (case insensitive)" << endl;
    }

    return 0;
}
```

4. Input Validation

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

int main() {
    int pilihan;

    cout << "Pilih menu (1-3): ";
    cin >> pilihan;

    // ? Tidak handle input invalid
    if (pilihan == 1) {
        cout << "Menu 1" << endl;
    } else if (pilihan == 2) {
        cout << "Menu 2" << endl;
    } else if (pilihan == 3) {
        cout << "Menu 3" << endl;
    }
```

```
    // Bagaimana jika user input 5 atau huruf?

    // ? Handle semua kemungkinan
    if (pilihan >= 1 && pilihan <= 3) {
        if (pilihan == 1) {
            cout << "Menu 1" << endl;
        } else if (pilihan == 2) {
            cout << "Menu 2" << endl;
        } else {
            cout << "Menu 3" << endl;
        }
    } else {
        cout << "Pilihan tidak valid! Harus 1-3." << endl;
    }

    return 0;
}
```

5. Boundary Values

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

int main() {
    int umur;
    cout << "Masukkan umur: ";
    cin >> umur;
```

```
    // Cek edge cases untuk umur
    if (umur < 0) {
        cout << "Error: Umur tidak boleh negatif!" << endl;
    } else if (umur == 0) {
        cout << "Bayi baru lahir" << endl;
    } else if (umur >= 1 && umur <= 12) {
        cout << "Anak-anak" << endl;
    } else if (umur >= 13 && umur <= 17) {
        cout << "Remaja" << endl;
    } else if (umur >= 18 && umur <= 59) {
        cout << "Dewasa" << endl;
    } else if (umur >= 60 && umur <= 120) {
        cout << "Lansia" << endl;
    } else {
        cout << "Error: Umur tidak realistis (>120)!" << endl;
    }

    return 0;
}
```

6. Assignment vs Comparison

```
int x = 5;

// ? Assignment dalam if (selalu true jika x != 0)
if (x = 10) {  // Ini assignment, bukan comparison!
    cout << "x sekarang = " << x << endl;  // x berubah jadi 10
```

```
}

// ? Comparison
if (x == 10) {
    cout << "x sama dengan 10" << endl;
}
```

Best Practices & Code Style

Slide 10

- **1. Consistent Indentation & Formatting**

```
// ? Sulit dibaca
if(a>b){
cout<<"a lebih besar";
}else{
cout<<"b lebih besar";
}

// ? Mudah dibaca
if (a > b) {
    cout << "a lebih besar" << endl;
} else {
    cout << "b lebih besar" << endl;
}
```

- **2. Meaningful Variable Names**

```
// ? Tidak jelas
int x, y, z;
```

```
if (x > 18 && y < 100000 && z == 1) {
    cout << "Approved" << endl;
}

// ? Jelas dan mudah dipahami
int umur, gaji, statusKerja;
if (umur > 18 && gaji < 100000 && statusKerja == 1) {
    cout << "Pinjaman disetujui" << endl;
}
```

- **3. Use Constants for Magic Numbers**

```
// ? Magic numbers
if (umur >= 17 && umur <= 60 && gaji >= 5000000) {
    cout << "Memenuhi syarat" << endl;
}

// ? Named constants
const int UMUR_MINIMAL = 17;
const int UMUR_MAKSIMAL = 60;
const int GAJI_MINIMAL = 5000000;

if (umur >= UMUR_MINIMAL && umur <= UMUR_MAKSIMAL && gaji >= GAJI_MINIMAL) {
    cout << "Memenuhi syarat" << endl;
}
```

- **4. Avoid Deep Nesting**

```
// ? Nesting terlalu dalam
if (userLoggedIn) {
    if (hasPermission) {
        if (dataValid) {
            if (connectionActive) {
                // proses data
            } else {
                cout << "Koneksi tidak aktif" << endl;
            }
        } else {
            cout << "Data tidak valid" << endl;
        }
    } else {
        cout << "Tidak ada permission" << endl;
    }
} else {
    cout << "User belum login" << endl;
}

// ? Early return/guard clauses
if (!userLoggedIn) {
    cout << "User belum login" << endl;
    return 0;
}

if (!hasPermission) {
```

```
    cout << "Tidak ada permission" << endl;
    return 0;
}

if (!dataValid) {
    cout << "Data tidak valid" << endl;
    return 0;
}

if (!connectionActive) {
    cout << "Koneksi tidak aktif" << endl;
    return 0;
}

// proses data (clean and simple)
```

- **5. Comprehensive Error Handling**

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

int main() {
    int pilihan;

    cout << "Masukkan pilihan (1-5): ";
```

```
    // Handle input failure
    if (!(cin >> pilihan)) {
        cout << "Error: Input harus berupa angka!" << endl;
        cin.clear(); // Clear error flags
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear buffer
        return 1;
    }

    // Validate range
    if (pilihan < 1 || pilihan > 5) {
        cout << "Error: Pilihan harus antara 1-5!" << endl;
        return 1;
    }

    // Process valid input
    switch (pilihan) {
        case 1:
            cout << "Anda memilih menu 1" << endl;
            break;
        case 2:
            cout << "Anda memilih menu 2" << endl;
            break;
        // ... dan seterusnya
        default:
            cout << "Menu tidak tersedia" << endl;
    }

    return 0;
}
```

- **6. Comments untuk Logic Kompleks**

```
// Hitung diskon berdasarkan multiple criteria
if ((pembelian >= 1000000 && member) ||           // Member dengan pembelian besar
    (pembelian >= 2000000 && !member) ||          // Non-member dengan pembelian sangat besar
    (isSpecialEvent && pembelian >= 500000)) {    // Special event dengan pembelian minimal

    double diskon = calculateDiscount(pembelian, member, isSpecialEvent);
    cout << "Anda mendapat diskon: " << diskon << "%" << endl;
}
```

- **7. Function untuk Logic yang Kompleks**

```
// ? Pisahkan logic kompleks ke dalam functions
bool isEligibleForLoan(int umur, double gaji, bool hasDebt) {
    return (umur >= 21 && umur <= 55) &&
           (gaji >= 3000000) &&
           (!hasDebt);
}

int main() {
    if (isEligibleForLoan(umur, gaji, hasDebt)) {
        cout << "Pinjaman disetujui" << endl;
    }
    return 0;
}
```

Latihan 1: Cek Bilangan Positif/Negatif/Nol (Easy)

Slide 11

Deskripsi: Buatlah program yang meminta user memasukkan sebuah bilangan, kemudian program akan mengecek dan menampilkan apakah bilangan tersebut positif, negatif, atau nol.

Input:

- Satu bilangan integer

Output yang Diharapkan:

- "Bilangan positif" jika > 0
- "Bilangan negatif" jika < 0
- "Bilangan nol" jika = 0

Contoh Run:

```
Masukkan bilangan: 5
Output: Bilangan positif
```

```
Masukkan bilangan: -3
Output: Bilangan negatif

Masukkan bilangan: 0
Output: Bilangan nol
```

Template Kode:

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

int main() {
    int bilangan;

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

    // TODO: Tulis kondisi untuk cek positif/negatif/nol

    return 0;
}
```

Konsep yang Dipelajari:

- Basic if-else statement

- Comparison operators (`>`, `<`, `==`)
- Simple decision making

Tips:

- Gunakan if-else if-else structure
- Perhatikan urutan kondisi yang logis
- Pastikan semua kemungkinan tercakup

Latihan 2: Kalkulator Sederhana (Easy-Medium)

Slide 12

Deskripsi: Buatlah program kalkulator sederhana yang dapat melakukan operasi dasar (+, -, *, /) dengan 2 bilangan. Program harus dapat menangani division by zero.

Input:

- Bilangan pertama (double)
- Operator (+, -, *, /)
- Bilangan kedua (double)

Output yang Diharapkan:

- Hasil operasi jika valid
- Pesan error jika division by zero

Contoh Run:

```
Masukkan bilangan pertama: 10
Masukkan operator (+, -, *, /): /
Masukkan bilangan kedua: 2
Hasil: 10 / 2 = 5

Masukkan bilangan pertama: 8
Masukkan operator (+, -, *, /): /
Masukkan bilangan kedua: 0
Error: Tidak dapat membagi dengan nol!
```

Template Kode:

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

int main() {
    double bil1, bil2;
    char op;

    cout << "Masukkan bilangan pertama: ";
    cin >> bil1;
    cout << "Masukkan operator (+, -, *, /): ";
    cin >> op;
    cout << "Masukkan bilangan kedua: ";
    cin >> bil2;
```

```
    // TODO: Implementasikan kalkulator dengan if-else

    return 0;
}
```

Konsep yang Dipelajari:

- Multiple conditions dengan else-if
- Character comparison
- Division by zero handling
- Formatting output

Tips:

- Gunakan else-if untuk setiap operator
- Cek division by zero sebelum melakukan pembagian
- Tampilkan hasil dengan format yang rapi

Latihan 3: Sistem Grading dengan Bonus (Medium)

Slide 13

Deskripsi: Buatlah sistem pemberian nilai (grading) yang tidak hanya berdasarkan nilai ujian, tetapi juga mempertimbangkan kehadiran. Ada bonus untuk kehadiran tinggi.

Kriteria:

- Nilai ujian 90-100: A
- Nilai ujian 80-89: B
- Nilai ujian 70-79: C
- Nilai ujian 60-69: D
- Nilai ujian <60: F

Bonus Kehadiran:

- Kehadiran ≥95%: Naik 1 tingkat (D→C, C→B, B→A)
- Kehadiran <75%: Turun 1 tingkat (A→B, B→C, C→D)

Contoh Run:

```
Masukkan nilai ujian (0-100): 78
Masukkan kehadiran (%): 96
Grade dasar: C
Bonus kehadiran: Naik 1 tingkat
Grade final: B

Masukkan nilai ujian (0-100): 85
Masukkan kehadiran (%): 70
Grade dasar: B
Penalty kehadiran: Turun 1 tingkat
Grade final: C
```

Template Kode:

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

int main() {
    double nilai;
    int kehadiran;

    cout << "Masukkan nilai ujian (0-100): ";
    cin >> nilai;
    cout << "Masukkan kehadiran (%): ";
```

```
    cin >> kehadiran;

    // TODO: Implementasikan sistem grading dengan bonus

    return 0;
}
```

Konsep yang Dipelajari:

- Nested conditionals
- Grade modification logic
- Complex decision making
- Multiple criteria evaluation

Tips:

- Tentukan grade dasar dulu, baru aplikasikan bonus/penalty
- Perhatikan batas atas dan bawah grade (F tidak bisa turun, A tidak bisa naik)
- Berikan feedback yang informatif kepada user

Latihan 4: Validasi Tanggal (Medium)

Slide 14

Deskripsi: Buatlah program untuk memvalidasi apakah tanggal yang dimasukkan user valid atau tidak. Program harus mempertimbangkan bulan dengan jumlah hari yang berbeda dan tahun kabisat.

Aturan:

- Bulan 1,3,5,7,8,10,12: 31 hari
- Bulan 4,6,9,11: 30 hari
- Bulan 2: 28 hari (29 jika tahun kabisat)
- Tahun kabisat: habis dibagi 4, TAPI jika habis dibagi 100 maka harus habis dibagi 400

Contoh Run:

```
Masukkan tanggal (dd mm yyyy): 29 2 2020
Tanggal valid (2020 adalah tahun kabisat)

Masukkan tanggal (dd mm yyyy): 29 2 2021
Tanggal tidak valid (2021 bukan tahun kabisat)
```

```
Masukkan tanggal (dd mm yyyy): 31 4 2023
Tanggal tidak valid (April hanya 30 hari)
```

Template Kode:

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

int main() {
    int hari, bulan, tahun;

    cout << "Masukkan tanggal (dd mm yyyy): ";
    cin >> hari >> bulan >> tahun;

    // TODO: Implementasikan validasi tanggal

    return 0;
}
```

Konsep yang Dipelajari:

- Complex nested conditions
- Logical operators (&&, ||)
- Multiple validation criteria

- Real-world problem solving

Tips:

- Validasi bulan dulu (1-12)
- Tentukan maksimal hari per bulan
- Untuk Februari, cek tahun kabisat
- Rumus tahun kabisat: `(tahun % 4 == 0 && tahun % 100 != 0) || (tahun % 400 == 0)`

Latihan 5: Sistem Diskon Belanja (Medium)

Slide 15

Deskripsi: Buatlah sistem perhitungan diskon untuk toko online dengan berbagai kriteria diskon yang bisa dikombinasikan.

Kriteria Diskon:

1. **Member Discount**: Member mendapat 10% diskon
2. **Quantity Discount**:
    - Beli ≥10 item: 5% tambahan
    - Beli ≥20 item: 10% tambahan
3. **Purchase Amount Discount**:
    - Belanja ≥Rp500,000: 5% tambahan
    - Belanja ≥Rp1,000,000: 10% tambahan
4. **Special Day**: Hari spesial (weekend) 15% diskon menggantikan member discount

Contoh Run:

```
Masukkan harga per item: 50000
Masukkan jumlah item: 15
Apakah member? (1=Ya, 0=Tidak): 1
Apakah hari spesial? (1=Ya, 0=Tidak): 0

=== DETAIL PEMBELIAN ===
Harga per item: Rp 50,000
Jumlah item: 15
Subtotal: Rp 750,000

=== DISKON ===
Member discount (10%): Rp 75,000
Quantity discount (5%): Rp 37,500
Amount discount (5%): Rp 37,500
Total diskon: Rp 150,000

=== TOTAL ===
Subtotal: Rp 750,000
Total diskon: Rp 150,000
Total bayar: Rp 600,000
```

Template Kode:

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

```
int main() {
    double hargaPerItem;
    int jumlahItem;
    bool isMember, isSpecialDay;

    cout << "Masukkan harga per item: ";
    cin >> hargaPerItem;
    cout << "Masukkan jumlah item: ";
    cin >> jumlahItem;
    cout << "Apakah member? (1=Ya, 0=Tidak): ";
    cin >> isMember;
    cout << "Apakah hari spesial? (1=Ya, 0=Tidak): ";
    cin >> isSpecialDay;

    // TODO: Implementasikan sistem diskon

    return 0;
}
```

Konsep yang Dipelajari:

- Multiple discount criteria
- Conditional discount stacking
- Boolean variables
- Complex business logic

- Detailed output formatting

Tips:

- Hitung subtotal dulu
- Terapkan diskon satu per satu
- Special day discount menggantikan, bukan menambah member discount
- Tampilkan breakdown diskon untuk transparency

Latihan 6: Game Tebak Angka dengan Clue (Medium)

Slide 16

Deskripsi: Buatlah game tebak angka di mana komputer memilih angka rahasia (1-100), dan user mendapat 3 kesempatan menebak dengan clue yang semakin spesifik.

Aturan:

- Angka rahasia: random 1-100 (untuk latihan, set manual)
- User punya 3 kesempatan
- Berikan clue setelah setiap tebakan yang salah
- Clue semakin spesifik setelah kesempatan berkurang

Contoh Run:

```
=== GAME TEBAK ANGKA ===
Saya memikirkan angka 1-100. Anda punya 3 kesempatan!

Kesempatan 1/3
Tebakan Anda: 50
Terlalu kecil! Angka saya lebih besar dari 50.
```

```
Kesempatan 2/3
Tebakan Anda: 80
Terlalu besar! Angka saya antara 50-80.

Kesempatan 3/3
Tebakan Anda: 65
Selamat! Anda menebak dengan benar! Angka saya adalah 65.

--- ATAU ---

Kesempatan 3/3
Tebakan Anda: 70
Maaf, kesempatan habis! Angka saya adalah 65.
```

Template Kode:

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

int main() {
    int angkaRahasia = 65; // Untuk latihan, set manual
    int tebakan;
    int kesempatan = 3;
    int batasAtas = 100, batasBawah = 1;

    cout << "=== GAME TEBAK ANGKA ===" << endl;
```

```
    cout << "Saya memikirkan angka 1-100. Anda punya 3 kesempatan!" << endl;

    // TODO: Implementasikan game logic

    return 0;
}
```

Konsep yang Dipelajari:

- Loop simulation with counters
- Progressive hint system
- Range tracking
- Game state management
- User experience design

Tips:

- Update batas atas/bawah setelah setiap tebakan
- Berikan clue yang semakin spesifik
- Handle win/lose conditions

- Tampilkan progress kesempatan yang tersisa

Latihan 7: Password Strength Checker (Medium-Hard)

Slide 17

Deskripsi: Buatlah program untuk mengecek kekuatan password berdasarkan beberapa kriteria dan memberikan skor serta rekomendasi perbaikan.

Kriteria Password (masing-masing +20 poin):

1. Panjang ≥8 karakter
2. Mengandung huruf kecil (a-z)
3. Mengandung huruf besar (A-Z)
4. Mengandung angka (0-9)
5. Mengandung simbol (!@#$%^&*)

Kategori Kekuatan:

- 0-40: Sangat Lemah
- 41-60: Lemah

- 61-80: Sedang
- 81-100: Kuat

Contoh Run:

```
Masukkan password: hello123
=== ANALISIS PASSWORD ===
Password: hello123

? Panjang ?8 karakter (8) [+20 poin]
? Mengandung huruf kecil [+20 poin]
? Tidak mengandung huruf besar [+0 poin]
? Mengandung angka [+20 poin]
? Tidak mengandung simbol [+0 poin]

Total Skor: 60/100
Kategori: Lemah

=== REKOMENDASI ===
- Tambahkan huruf besar (A-Z)
- Tambahkan simbol (!@#$%^&*)
```

Template Kode:

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

int main() {
    string password;
    int skor = 0;

    cout << "Masukkan password: ";
    getline(cin, password);

    cout << "\n=== ANALISIS PASSWORD ===" << endl;
    cout << "Password: " << password << endl << endl;

    // TODO: Implementasikan password strength checker

    return 0;
}
```

Konsep yang Dipelajari:

- String analysis
- Character type checking
- Multiple condition evaluation

- Scoring system
- User feedback generation

Tips:

- Gunakan ASCII values untuk check character types
- Huruf kecil: 97-122 ('a'-'z')
- Huruf besar: 65-90 ('A'-'Z')
- Angka: 48-57 ('0'-'9')
- Loop through setiap karakter password
- Berikan feedback yang konstruktif

Latihan 8: ATM Transaction Simulator (Hard)

Slide 18

Deskripsi: Buatlah simulator transaksi ATM yang lengkap dengan validasi PIN, cek saldo, dan berbagai jenis transaksi dengan limit dan biaya admin.

Fitur:

1. **Login**: PIN harus benar (3 kali kesempatan)
2. **Cek Saldo**: Gratis
3. **Tarik Tunai**:
    - Minimum Rp50,000
    - Maksimum Rp2,500,000 per transaksi
    - Biaya admin Rp6,500
4. **Transfer**:
    - Minimum Rp10,000

- Biaya admin Rp2,500 (sesama bank), Rp6,500 (beda bank)

Contoh Run:

```
=== SELAMAT DATANG DI ATM ===
Masukkan PIN: 1234

? PIN Benar! Selamat datang.
Saldo Anda: Rp 1,500,000

Menu:
1. Cek Saldo
2. Tarik Tunai
3. Transfer
4. Keluar

Pilih menu: 2
Masukkan jumlah: 200000

=== KONFIRMASI PENARIKAN ===
Jumlah penarikan: Rp 200,000
Biaya admin: Rp 6,500
Total debet: Rp 206,500
Saldo tersisa: Rp 1,293,500

Konfirmasi? (y/n): y
? Transaksi berhasil!
```

```
Ambil uang Anda: Rp 200,000
Saldo tersisa: Rp 1,293,500
```

Template Kode:

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

int main() {
    const int PIN_BENAR = 1234;
    double saldo = 1500000; // Saldo awal
    int pin, pilihan;
    int kesempatanPin = 3;

    cout << "=== SELAMAT DATANG DI ATM ===" << endl;

    // TODO: Implementasikan ATM simulator

    return 0;
}
```

Konsep yang Dipelajari:

- Multi-step authentication
- Complex transaction logic

- Input validation
- State management
- Real-world business rules
- Error handling

Tips:

- Implementasikan login dulu sebelum menu utama
- Validasi setiap input (jumlah, PIN, pilihan menu)
- Cek saldo mencukupi sebelum transaksi
- Berikan konfirmasi sebelum eksekusi
- Handle edge cases (saldo kurang, limit terlampaui)

Latihan 9: Smart Traffic Light Controller (Hard)

Slide 19

Deskripsi: Buatlah simulator sistem traffic light pintar yang menyesuaikan waktu lampu berdasarkan kondisi lalu lintas, waktu, dan prioritas kendaraan darurat.

Input Kondisi:

- Jam (0-23)
- Jumlah kendaraan arah Utara-Selatan
- Jumlah kendaraan arah Timur-Barat
- Ada ambulans/pemadam? (boolean)
- Cuaca (1=Cerah, 2=Hujan, 3=Kabut)

Logika Traffic Light:

1. **Emergency**: Jika ada ambulans → langsung hijau arah ambulans (30 detik)
2. **Time-based**:

- Rush hour (7-9, 17-19): Waktu lebih lama
- Normal hour: Waktu standar
- Night time (22-5): Waktu lebih pendek

3. **Traffic-based**: Arah dengan kendaraan lebih banyak dapat waktu lebih lama
4. **Weather adjustment**: Hujan/kabut → tambah 10 detik

Contoh Run:

```
=== SMART TRAFFIC LIGHT CONTROLLER ===
Masukkan jam (0-23): 8
Masukkan jumlah kendaraan Utara-Selatan: 25
Masukkan jumlah kendaraan Timur-Barat: 15
Ada kendaraan darurat? (1=Ya, 0=Tidak): 0
Cuaca (1=Cerah, 2=Hujan, 3=Kabut): 2

=== ANALISIS KONDISI ===
? Waktu: 08:00 (Rush Hour)
? Kendaraan U-S: 25, T-B: 15
?? Cuaca: Hujan
? Emergency: Tidak ada

=== PENGATURAN LAMPU ===
```

```
Prioritas: Utara-Selatan (lebih ramai)
Base time: 60 detik (rush hour)
Traffic bonus: +10 detik (selisih 10 kendaraan)
Weather adjustment: +10 detik (hujan)

?? HASIL PENGATURAN ??
? Hijau Utara-Selatan: 80 detik
? Kuning: 5 detik
? Hijau Timur-Barat: 50 detik
? Kuning: 5 detik

Total cycle: 140 detik
```

Template Kode:

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

int main() {
    int jam, kendaraanUS, kendaraanTB, cuaca;
    bool emergency;

    cout << "=== SMART TRAFFIC LIGHT CONTROLLER ===" << endl;
    cout << "Masukkan jam (0-23): ";
    cin >> jam;
    cout << "Masukkan jumlah kendaraan Utara-Selatan: ";
    cin >> kendaraanUS;
```

```
    cout << "Masukkan jumlah kendaraan Timur-Barat: ";
    cin >> kendaraanTB;
    cout << "Ada kendaraan darurat? (1=Ya, 0=Tidak): ";
    cin >> emergency;
    cout << "Cuaca (1=Cerah, 2=Hujan, 3=Kabut): ";
    cin >> cuaca;

    // TODO: Implementasikan smart traffic light logic

    return 0;
}
```

Konsep yang Dipelajari:

- Multi-variable decision making
- Priority-based logic
- Time-based conditions
- Complex algorithm design
- Real-world system simulation

Tips:

- Handle emergency case terlebih dahulu

- Tentukan base time berdasarkan jam
- Hitung traffic difference untuk bonus time
- Apply weather adjustment
- Berikan output yang informatif dan visual

Latihan 10: University Admission System (Hard)

Slide 20

Deskripsi: Buatlah sistem penerimaan mahasiswa universitas yang komprehensif dengan multiple criteria, quota system, dan dynamic scoring.

Kriteria Penerimaan:

1. **Academic Score** (40%):
    - Matematika (wajib ≥70)
    - Bahasa Inggris (wajib ≥65)
    - IPA/IPS (sesuai jurusan, wajib ≥70)
2. **Additional Factors** (60%):
    - Prestasi akademik: +10 poin
    - Prestasi non-akademik: +5 poin

- Ekonomi kurang mampu: +8 poin
- Daerah 3T (Terdepan, Terpencil, Tertinggal): +12 poin

3. **Jurusan & Quota**:
   - Teknik: Min score 85, quota 50
   - Kedokteran: Min score 90, quota 25
   - Ekonomi: Min score 75, quota 75
   - Sastra: Min score 70, quota 100

Contoh Run:

```
=== SISTEM PENERIMAAN UNIVERSITAS ===

Data Calon Mahasiswa:
Nama: Ahmad Fauzi
Matematika: 85
Bahasa Inggris: 80
IPA: 88
Jurusan pilihan: TEKNIK
```

```
Prestasi akademik (1=Ya, 0=Tidak): 1
Prestasi non-akademik (1=Ya, 0=Tidak): 0
Ekonomi kurang mampu (1=Ya, 0=Tidak): 1
Dari daerah 3T (1=Ya, 0=Tidak): 0

=== EVALUASI PENERIMAAN ===
Nama: Ahmad Fauzi
Jurusan: TEKNIK

? Syarat Minimum:
- Matematika: 85 ? 70 ?
- Bahasa Inggris: 80 ? 65 ?
- IPA: 88 ? 70 ?

? Perhitungan Skor:
Academic Score (40%):
- Rata-rata: (85+80+88)/3 = 84.33
- Academic component: 84.33 × 0.4 = 33.73

Additional Factors (60%):
- Base score: 60
- Prestasi akademik: +10
- Ekonomi kurang mampu: +8
- Additional component: (60+10+8) × 0.6 = 46.8

? TOTAL SCORE: 33.73 + 46.8 = 80.53

? HASIL: TIDAK DITERIMA
Alasan: Skor total (80.53) < minimum Teknik (85)
```

```
? Rekomendasi:
- Tingkatkan nilai akademik
- Cari prestasi non-akademik (+5 poin)
- Pertimbangkan jurusan Ekonomi (min. 75)
```

Template Kode:

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

int main() {
    string nama, jurusan;
    double matematika, bahasaInggris, ipa;
    bool prestasiAkademik, prestasiNonAkademik, ekonomiKurangMampu, daerah3T;

    cout << "=== SISTEM PENERIMAAN UNIVERSITAS ===" << endl << endl;

    cout << "Data Calon Mahasiswa:" << endl;
    cout << "Nama: ";
    getline(cin, nama);
    cout << "Matematika: ";
    cin >> matematika;
    cout << "Bahasa Inggris: ";
    cin >> bahasaInggris;
    cout << "IPA: ";
```

```
    cin >> ipa;
    cout << "Jurusan pilihan (TEKNIK/KEDOKTERAN/EKONOMI/SASTRA): ";
    cin >> jurusan;

    cout << "\nPrestasi akademik (1=Ya, 0=Tidak): ";
    cin >> prestasiAkademik;
    cout << "Prestasi non-akademik (1=Ya, 0=Tidak): ";
    cin >> prestasiNonAkademik;
    cout << "Ekonomi kurang mampu (1=Ya, 0=Tidak): ";
    cin >> ekonomiKurangMampu;
    cout << "Dari daerah 3T (1=Ya, 0=Tidak): ";
    cin >> daerah3T;

    // TODO: Implementasikan sistem penerimaan universitas

    return 0;
}
```

Konsep yang Dipelajari:

- Complex multi-criteria evaluation
- Weighted scoring system
- Quota management
- Comprehensive validation

- Professional system design
- Detailed reporting

Tips:

- Cek syarat minimum dulu sebelum hitung skor
- Implementasikan weighted scoring dengan benar
- Handle berbagai jurusan dengan kriteria berbeda
- Berikan feedback yang konstruktif dan actionable
- Tampilkan breakdown perhitungan untuk transparency