

- Topik Bahasan:
 - Pengenalan Fungsi
 - Fungsi Void
 - Fungsi dengan Return
 - Multiple Parameters
 - Scope Variable
 - Latihan

- **Problem: Kode Berulang**

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

int main() {
    // Mencetak blok kode yang sama di tempat yang berbeda
    cout << "======" << endl;
    cout << "    PROGRAM KALKULATOR    " << endl;
    cout << "======" << endl;

    // ... kode program ...

    // Mencetak lagi di bagian lain
    cout << "======" << endl;
    cout << "    PROGRAM KALKULATOR    " << endl;
    cout << "======" << endl;

    return 0;
}
```

- **Masalah:**

- Kode tidak efisien (copy-paste berulang)
- Sulit maintenance (jika ada perubahan, harus edit di banyak tempat)

Apa itu Fungsi?

- **Definisi:**

- Fungsi adalah blok kode yang dapat dipanggil berkali-kali
- Memiliki tugas spesifik yang terdefinisi dengan jelas
- Dapat menerima input (parameter) dan mengembalikan output (return)

Keuntungan Menggunakan Fungsi

- **1. Reusable (Dapat Digunakan Berulang)**

- Tulis sekali, pakai berkali-kali
- Tidak perlu copy-paste kode

- **2. Readable (Mudah Dibaca)**

- Kode lebih terstruktur dan rapi
- Nama fungsi menjelaskan apa yang dilakukan
- Lebih mudah dipahami oleh programmer lain

- **3. Maintainable (Mudah Dipelihara)**

- Jika ada bug, cukup perbaiki di satu tempat
- Update fitur lebih mudah

- Testing lebih mudah

- **4. Modular**

- Program dipecah menjadi bagian-bagian kecil
- Setiap bagian punya tanggung jawab spesifik
- Lebih mudah dikembangkan tim

- **Syntax Fungsi:**

```
tipeReturn namaFungsi(parameter1, parameter2, ...) {  
    // Body fungsi  
    // Kode yang akan dijalankan  
  
    return nilai; // jika ada return  
}
```

- **Komponen Fungsi:**

- **tipeReturn:** Tipe data yang dikembalikan (void, int, double, bool, string, dll)
- **namaFungsi:** Nama fungsi (gunakan camelCase atau snake_case)
- **parameter:** Input yang diterima fungsi (opsional)
- **body:** Kode yang dieksekusi
- **return:** Nilai yang dikembalikan (jika bukan void)

- **Contoh:**

```
void salam() {  
    cout << "Halo, Selamat Datang!" << endl;  
}  
  
int tambah(int a, int b) {  
    return a + b;  
}
```

Fungsi Void

- **Apa itu Fungsi Void?**

- Fungsi yang tidak mengembalikan nilai
- Hanya menjalankan aksi/perintah
- Menggunakan keyword `void` sebagai tipe return

- **Karakteristik:**

- Tidak ada statement `return` (atau `return;` tanpa nilai)
- Biasanya untuk: mencetak output, mengubah data, menjalankan proses

- **Syntax Dasar:**

```
void namaFungsi() {  
    // kode yang akan dijalankan  
}
```

- **Kapan Menggunakan Void?**

- Untuk aksi yang tidak perlu hasil balik
- Contoh: cetak pesan, tampilkan menu

Fungsi Void Tanpa Parameter

Slide 7

- **Fungsi tanpa input:**

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

// Deklarasi fungsi
void salam() {
    cout << "Halo, Selamat Datang!" << endl;
    cout << "Selamat belajar C++" << endl;
}

void cetakGaris() {
    cout << "======" << endl;
}

int main() {
    // Memanggil fungsi
    salam();
    cetakGaris();

    return 0;
}
```

- **Output:**

```
Halo, Selamat Datang!  
Selamat belajar C++  
=====
```

- **Catatan:** Fungsi harus dideklarasikan sebelum dipanggil (di atas main)

- **Fungsi dengan input:**

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

// Fungsi dengan 1 parameter
void sapa(string nama) {
    cout << "Halo, " << nama << "!" << endl;
    cout << "Selamat datang di program kami" << endl;
}

// Fungsi dengan 2 parameter
void tampilNilai(string matakuliah, int nilai) {
    cout << "Mata Kuliah: " << matakuliah << endl;
    cout << "Nilai: " << nilai << endl;
}

int main() {
    sapa("Budi");
    sapa("Ani");

    cout << endl;
```

```
tampilNilai("Dasar Pemrograman", 85);
tampilNilai("Matematika Diskrit", 90);

return 0;
}
```

- **Apa itu Fungsi Return?**

- Fungsi yang mengembalikan nilai ke pemanggil
- Menggunakan tipe data spesifik (int, double, bool, string, dll)
- Wajib ada statement `return`

- **Karakteristik:**

- Harus mengembalikan nilai sesuai tipe data
- Nilai return dapat ditampung dalam variabel
- Dapat langsung digunakan dalam operasi/ekspresi

- **Syntax Dasar:**

```
tipeData namaFungsi(parameter) {  
    // kode  
    return nilai; // nilai harus sesuai tipe data  
}
```

- **Kapan Menggunakan Return?**

- Ketika butuh hasil perhitungan
- Ketika butuh nilai untuk diproses lebih lanjut
- Contoh: kalkulasi, pencarian, validasi

- **3 Cara Menggunakan Return Value:**

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

int tambah(int a, int b) {
    return a + b;
}

int main() {
    // 1. Simpan dalam variabel
    int hasil = tambah(10, 20);
    cout << "Hasil: " << hasil << endl;

    // 2. Langsung tampilkan
    cout << "Hasil: " << tambah(15, 25) << endl;

    // 3. Gunakan dalam ekspresi/operasi
    int total = tambah(5, 10) + tambah(3, 7);
    cout << "Total: " << total << endl;

    // 4. Gunakan dalam kondisi
    if (tambah(2, 3) > 4) {
```

```
    cout << "Lebih dari 4" << endl;
}

return 0;
}
```

- **Apa itu Scope?**

- Scope = ruang lingkup/area di mana variabel dapat diakses
- Menentukan di mana variabel bisa digunakan
- Penting untuk menghindari error dan conflict

- **Jenis Scope:**

- **Local Scope:** Variabel hanya bisa diakses dalam fungsi tempat dibuat
- **Global Scope:** Variabel bisa diakses di seluruh program (tidak disarankan)

- **Local Variable dalam Fungsi:**

```
void fungsiA() {  
    int x = 10; // x hanya ada di fungsiA  
    cout << x << endl;  
}
```

```
void fungsiB() {
    // cout << x << endl; // ERROR! x tidak dikenal di sini
    int y = 20; // y hanya ada di fungsiB
}

int main() {
    // cout << x << endl; // ERROR! x tidak dikenal di main
    // cout << y << endl; // ERROR! y tidak dikenal di main
    return 0;
}
```

Parameter = Local Variable

Slide 12

- Parameter adalah local variable:

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

void hitungLuas(int panjang, int lebar) {
    // panjang dan lebar adalah local variable
    int luas = panjang * lebar; // luas juga local
    cout << "Luas: " << luas << endl;
}

void hitungKeliling(int panjang, int lebar) {
    // ini adalah panjang dan lebar BERBEDA dari fungsi hitungLuas
    int keliling = 2 * (panjang + lebar);
    cout << "Keliling: " << keliling << endl;
}

int main() {
    hitungLuas(5, 3);
    hitungKeliling(5, 3);

    // ERROR! tidak bisa akses panjang, lebar, luas, keliling
    // cout << luas << endl; // ERROR!
```

```
    return 0;  
}
```

Bahaya Global Variable

Slide 13

- **Global Variable = variabel di luar semua fungsi:**

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

int nilai = 100; // GLOBAL - bisa diakses semua fungsi

void ubahNilai() {
    nilai = 50; // mengubah global variable
}

void tampilNilai() {
    cout << "Nilai: " << nilai << endl;
}

int main() {
    tampilNilai(); // Output: 100
    ubahNilai();
    tampilNilai(); // Output: 50 (berubah!)
    return 0;
}
```

- Kenapa Tidak Disarankan? Karena akan sulit untuk tracking ketika ada perubahan nilai

- **Saran:**

- Gunakan parameter untuk passing data
- Gunakan return untuk mengembalikan hasil
- Hindari global variable sebisa mungkin

- **Deskripsi Program:**

Buat program untuk mengelola nilai mahasiswa dengan fitur:

1. Input data mahasiswa (nama dan nilai)
2. Tampilkan semua data mahasiswa
3. Hitung rata-rata nilai kelas
4. Cari mahasiswa dengan nilai tertinggi
5. Tampilkan mahasiswa yang lulus (nilai ≥ 60)

- **Requirement:**

- Gunakan **vector** untuk menyimpan nama dan nilai
- Buat **fungsi terpisah** untuk setiap fitur
- Gunakan **looping** untuk input dan proses data

- Gunakan **kondisi** untuk validasi dan filter data
- Gunakan **menu dengan switch-case**

- **Fungsi yang harus dibuat:**

- `void tampilMenu()` - menampilkan menu
- `void inputData(vector<string> &nama, vector<int> &nilai)` - input data
- `void tampilData(vector<string> nama, vector<int> nilai)` - tampilkan semua
- `double hitungRataRata(vector<int> nilai)` - hitung rata-rata
- `void cariTertinggi(vector<string> nama, vector<int> nilai)` - cari nilai max
- `void tampilLulus(vector<string> nama, vector<int> nilai)` - filter lulus

Sample Output yang Diharapkan

Slide 15

```
=====
SISTEM MANAJEMEN NILAI MAHASISWA
=====
```

MENU:

1. Input Data Mahasiswa
2. Tampilkan Semua Data
3. Hitung Rata-rata Nilai Kelas
4. Cari Mahasiswa Nilai Tertinggi
5. Tampilkan Mahasiswa yang Lulus
6. Exit

Pilih menu: 1

Berapa mahasiswa yang akan diinput? 4

Input data mahasiswa ke-1:

Nama: Ahmad Zaki

Nilai: 85

Input data mahasiswa ke-2:

Nama: Siti Nurhaliza

Nilai: 92

Input data mahasiswa ke-3:

Nama: Budi Santoso

Nilai: 55

Input data mahasiswa ke-4:

Nama: Dewi Lestari

Nilai: 78

Data berhasil disimpan!

Sample Output (Lanjutan)

Slide 16

Pilih menu: 2

DATA MAHASISWA:

No	Nama	Nilai
1	Ahmad Zaki	85
2	Siti Nurhaliza	92
3	Budi Santoso	55
4	Dewi Lestari	78

Pilih menu: 3

RATA-RATA NILAI KELAS: 77.5

Pilih menu: 4

MAHASISWA DENGAN NILAI TERTINGGI:

Nama: Siti Nurhaliza

Nilai: 92

Pilih menu: 5

MAHASISWA YANG LULUS (Nilai >= 60):

=====

1. Ahmad Zaki - Nilai: 85
2. Siti Nurhaliza - Nilai: 92
3. Dewi Lestari - Nilai: 78

=====

Total mahasiswa lulus: 3 dari 4

Pilih menu: 6

Terima kasih! Program selesai.