Tentu, mari kita bedah bersama.

Kebingungan Anda soal string di C sangat wajar. Ini adalah salah satu topik yang paling *tricky* bagi mahasiswa ilmu komputer saat pertama kali belajar C.

Masalah utamanya adalah: **C tidak punya tipe data string bawaan.**

Tidak seperti Python atau Java di mana Anda bisa langsung menulis string s = "Hello", di C, sebuah "string" hanyalah sebuah **aturan** atau **konvensi**.

👑 Aturan #1: String di C adalah Array char

Aturan paling dasarnya adalah:

String adalah sebuah array dari char yang diakhiri oleh karakter spesial \0 (null terminator).

\0 ini sangat penting. Inilah cara C mengetahui di mana sebuah string berakhir.

Lihat perbedaan ini:

- char arr[] = {'H', 'a', 'l', 'o'};
    - Ini **bukan** string. Ini hanyalah array berisi 4 karakter.
- char str[] = {'H', 'a', 'l', 'o', '\0'};
    - Ini **adalah** string C yang valid.
- char str[] = "Halo";
    - Ini adalah cara singkat. Tanda kutip dua ("") secara otomatis menambahkan \0 di akhir.
    - Jadi, di memori, "Halo" sebenarnya disimpan sebagai 5 karakter: H, a, l, o, \0.

⌨️ Input Karakter vs. Input String

Ini adalah bagian yang sering membuat bingung.

1. Input Karakter Tunggal (char)

Anda ingin membaca *satu* huruf saja.

- **Fungsi:** scanf("%c", ...) atau getchar()
- **Contoh:** char pilihan; scanf("%c", &pilihan);

- **Masalah Umum:** scanf("%c") akan membaca *karakter apa saja*, termasuk spasi dan "Enter" (\n) yang mungkin tersisa dari scanf sebelumnya.
- **Solusi "Pro":** Selalu gunakan spasi sebelum %c.

C

// Spasi sebelum %c menyuruh scanf untuk "membuang"

// sisa whitespace (seperti '\n') sebelum membaca karakter baru.

scanf(" %c", &pilihan);

2. Input String (Array char)

Anda ingin membaca *kumpulan* huruf.

PENTING: Anda harus **menyediakan tempat** (array) terlebih dahulu.

C

// Siapkan "kotak" yang bisa menampung 50 karakter

// (Sebenarnya 49 karakter + 1 untuk '\0')

char nama[50];

Sekarang, ada dua cara utama untuk mengisinya:

Cara 1: scanf("%s", ...) (Cara Cepat, tapi BERBAHAYA)

- **Contoh:** scanf("%s", nama); (Perhatikan: **tidak pakai &** untuk array)
- **Kelebihan:** Mudah diingat.
- **Kekurangan:**
    1. **Berhenti di Spasi:** Jika Anda mengetik "Budi Setiawan", scanf hanya akan mengambil "Budi".
    2. **Tidak Aman (Buffer Overflow):** Jika nama hanya [10] dan Anda mengetik 20 karakter, program Anda akan merusak memori di sekitarnya. Ini adalah celah keamanan besar.
- **Versi Sedikit Lebih Aman:** scanf("%49s", nama); (angka 49 memberi batas, 1 sisa untuk \0). Tapi tetap berhenti di spasi.

Cara 2: fgets(...) (Cara yang Disarankan dan AMAN)

- **Fungsi:** fgets(variabel, ukuran_maksimal, stdin);
- **Contoh:** fgets(nama, 50, stdin);

- **Kelebihan:**
    1. **Aman:** Tidak akan melebihi ukuran array (50).
    2. **Membaca Spasi:** Akan membaca seluruh "Budi Setiawan" sampai Anda menekan Enter.
- **Satu "Kekurangan" Kecil:** fgets juga menyimpan karakter Enter (\n) di akhir string. Anda biasanya perlu menghapusnya secara manual.

⚙️ Operasi String (Pustaka <string.h>)

Karena string adalah array, Anda **tidak bisa** melakukan operasi seperti ini:

- string1 == string2 (SALAH! Ini membandingkan alamat memori, bukan isinya)
- string1 = string2 (SALAH! Anda tidak bisa menyalin array dengan =)

Anda **wajib** menggunakan fungsi dari *library* <string.h>.

#include <string.h>

1. **Mencari Panjang String: strlen()**
    - int panjang = strlen(nama);
    - Menghitung jumlah karakter *sebelum* \0.
    - strlen("Halo") akan menghasilkan 4.
2. **Membandingkan String: strcmp()** (String Compare)
    - if (strcmp(nama, "Budi") == 0)
    - Ini adalah cara yang benar untuk membandingkan apakah nama == "Budi".
    - Ia mengembalikan:
        - 0 jika string-nya **sama persis**.
        - < 0 jika string1 lebih kecil (duluan) secara alfabet.
        - > 0 jika string1 lebih besar (belakangan) secara alfabet.
3. **Menyalin String: strcpy()** (String Copy)
    - strcpy(tujuan, sumber);
    - **Contoh:** strcpy(nama, "Budi"); (Ini seperti nama = "Budi")
    - **Bahaya:** strcpy tidak aman! Jika tujuan terlalu kecil, ia akan menyebabkan *buffer overflow*. Gunakan strncpy atau snprintf untuk versi yang lebih aman.

4. **Menggabung String: strcat()** (String Concatenate)
    - strcat(tujuan, sumber);
    - **Contoh:**

C

char sapa[20] = "Halo ";

strcat(sapa, nama); // Jika nama="Budi", sapa akan menjadi "Halo Budi"

    - **Bahaya:** Sama seperti strcpy, strcat juga tidak aman. Pastikan array tujuan (yaitu sapa) cukup besar untuk menampung *gabungan* keduanya.

⚠️ **Rangkuman Aturan & Jebakan**

1. **Selalu ingat \0:** Saat membuat array char manual, pastikan ada ruang untuk \0 dan pastikan \0 ada di akhir.
2. **sizeof vs strlen:**
    - char str[50] = "Halo";
    - sizeof(str) adalah **50** (ukuran total array di memori).
    - strlen(str) adalah **4** (panjang string "Halo").
3. **Gunakan fgets untuk input:** Ini adalah praktik terbaik dan teraman.
4. **Gunakan strcmp untuk membandingkan:** Jangan pernah pakai == untuk membandingkan isi string.
5. **Gunakan <string.h>:** Jangan coba-coba memanipulasi string (menyalin/menggabung) secara manual kecuali Anda tahu persis apa yang Anda lakukan.

Semoga ini membantu menjernihkan kebingungan Anda! String di C memang rumit, tapi sekali Anda paham konsep intinya (array char + \0), semuanya akan jadi lebih masuk akal.

Apakah Anda ingin melihat contoh program lengkap yang menggunakan fgets dan strcmp untuk program login sederhana?