

VERSI 1.0  
AGUSTUS, 2024



# [PEMROGRAMAN DASAR]

MODUL 1 – INTRODUCTION, SYNTAX, INPUT/OUTPUT, COMMENTS,  
VARIABLES, DATA TYPES

DISUSUN OLEH:

WEMPY ADITYA WIRYAWAN

MUHAMMAD ZAKY DARAJAT

DIAUDIT OLEH:

- HARDIANTO WIBOWO, S.KOM, M.T

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

## [PEMROGRAMAN DASAR]

---

### PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

- Pengenalan Bahasa C
- Syntax
- Input/Output
- Comments
- Variables
- Data Types

---

### TUJUAN

- Mahasiswa dapat memahami sejarah, karakteristik, dan kegunaan bahasa pemrograman C sebagai dasar.
- Mahasiswa dapat memahami struktur dan aturan penulisan code agar dapat membuat program dengan sintaks yang benar.
- Mahasiswa dapat mengambil input dari pengguna dan menampilkan output ke layar.
- Mahasiswa dapat menggunakan komentar untuk memberikan penjelasan, dokumentasi, atau menyembunyikan bagian code tertentu.
- Mahasiswa dapat memahami konsep variabel dan cara mendeklarasikan serta menggunakan variabel untuk menyimpan dan memanipulasi data.
- Mahasiswa dapat memahami berbagai tipe data dan fungsinya dalam penyimpanan dan pengolahan data.

---

### TARGET MODUL

- Mahasiswa dapat menjelaskan latar belakang dan ciri khas bahasa C serta mengenal lingkungan pengembangan bahasa C.
- Mahasiswa dapat menulis program sederhana dengan menggunakan sintaks.

- Mahasiswa dapat membuat program yang menerima input, memprosesnya, dan menampilkan hasilnya kepada pengguna.
- Mahasiswa dapat menambahkan komentar pada code program dengan tepat dan mengerti fungsinya.
- Mahasiswa dapat mendeklarasikan variabel, menginisialisasi nilainya, dan melakukan operasi dasar pada variabel.
- Mahasiswa dapat menggunakan tipe data yang sesuai untuk menyimpan berbagai jenis nilai dan memahami batasan dan ukuran masing-masing tipe data.

---

## PERSIAPAN SOFTWARE/APLIKASI

- Komputer/Laptop
- Software IDE (Falcon/Dev C++)

---

## MATERI PRAKTIKUM

### INTRODUCTION

#### ➤ APA ITU BAHASA C

Bahasa C, sering disebut "C," adalah salah satu bahasa pemrograman yang memiliki sejarah yang kaya dan berperan penting dalam perkembangan dunia pemrograman. Dennis Ritchie, seorang ilmuwan komputer yang bekerja di Bell Telephone Laboratories Inc., mengembangkan bahasa ini pada tahun 1970-an. Tujuan utama pembuatan bahasa C adalah untuk membuat bahasa pemrograman yang efisien, portabel, dan dapat digunakan untuk mengembangkan sistem operasi UNIX.

Awalnya, bahasa C digunakan di komputer Digital Equipment Corporation PDP-11 yang menjalankan sistem operasi UNIX. Namun, seiring berjalannya waktu, bahasa C menjadi semakin populer dan digunakan di berbagai platform dan sistem operasi. Kemampuannya dalam membuat kode yang efisien dan portabel membuatnya menjadi pilihan utama dalam pengembangan sistem dan perangkat lunak.

Salah satu keunggulan utama dari bahasa C adalah kemampuannya sebagai bahasa pemrograman umum (*General-Purpose Programming Language*). Ini berarti bahwa bahasa C dapat digunakan untuk membuat berbagai jenis aplikasi, termasuk perangkat lunak desktop, sistem operasi, perangkat lunak embedded, dan banyak lagi. Fleksibilitas ini telah membuat bahasa C menjadi landasan penting dalam dunia pemrograman.

Bahasa C juga dikenal dengan sebutan "*God's Programming Language*" karena kekuatannya dan fleksibilitasnya. Meskipun pada awalnya, bahasa C mungkin tampak sedikit kompleks bagi pemula, banyak programmer menganggap memahami bahasa ini sebagai langkah penting dalam perjalanan belajar pemrograman. Memahami bahasa C membantu programmer memahami dasar-dasar pemrograman, seperti variabel, tipe data, operasi aritmatika, percabangan, perulangan, dan masih banyak lagi.

Keunikan sintaksis dan konsep dalam bahasa C juga memberikan landasan bagi bahasa-bahasa pemrograman modern lainnya. Misalnya, bahasa C++ dikembangkan sebagai ekstensi dari bahasa C, dengan penambahan fitur-fitur seperti objek dan kelas. Bahasa Java, yang digunakan secara luas dalam pengembangan aplikasi berbasis web dan perangkat lunak lainnya, juga terinspirasi oleh bahasa C dalam banyak aspek.

Dengan mempelajari bahasa C, tidak hanya akan memahami dasar-dasar pemrograman, tetapi juga akan membangun fondasi yang kuat untuk memahami bahasa pemrograman lainnya. Memahami konsep seperti alokasi memori, penggunaan pointer, dan pengelolaan sumber daya akan membantu kalian lebih baik dalam menghadapi tantangan dalam pengembangan perangkat lunak modern.

Dalam kesimpulannya, bahasa C adalah salah satu bahasa pemrograman yang paling penting dalam sejarah komputasi. Meskipun telah berusia beberapa dekade, warisannya masih terasa kuat di dunia pemrograman saat ini. Dengan memahami bahasa C, akan membuka pintu untuk pemahaman yang lebih mendalam tentang dunia pemrograman secara keseluruhan.

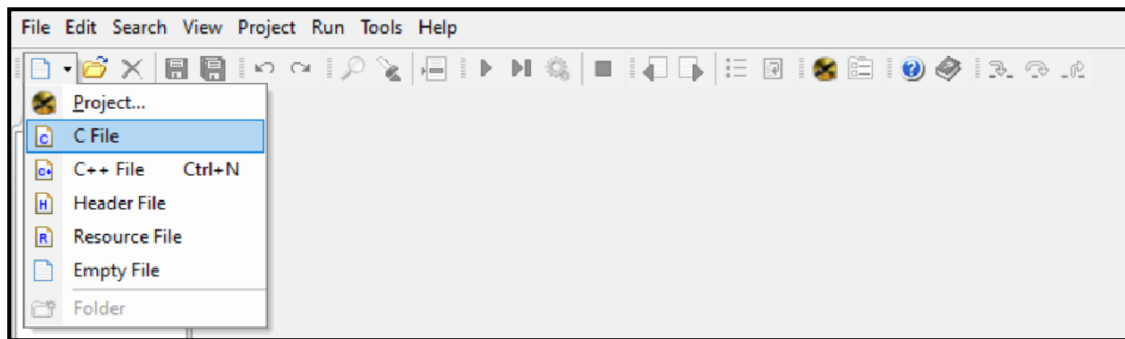
#### ➤ **INSTALASI IDE (Integrated Development Environment)**

IDE merupakan sebuah perangkat lunak yang menggabungkan berbagai alat dan fitur untuk memudahkan proses pengembangan perangkat lunak (*software development*) dalam suatu bahasa pemrograman tertentu. Untuk install Falcon C++/Dev C++ bisa mengakses link berikut:

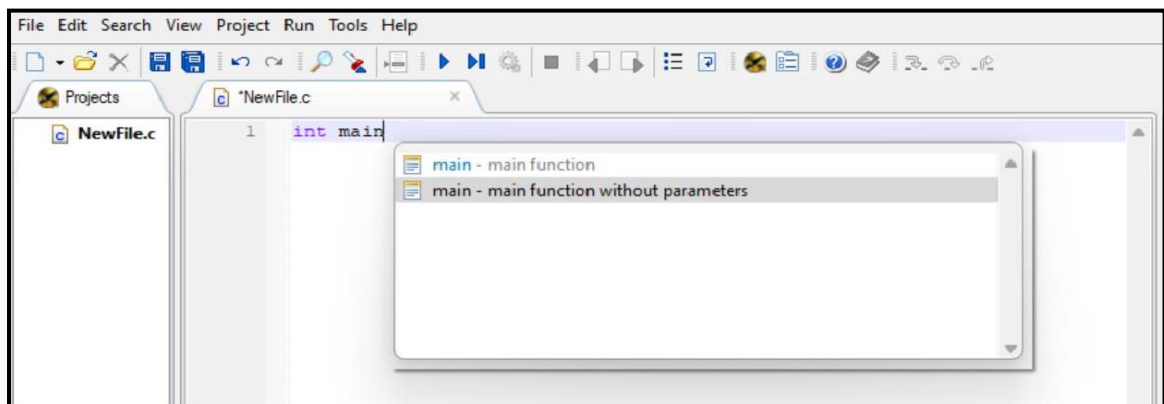
<https://drive.google.com/drive/folders/1h-tYM0ifZiJleP7GYupp4GjP0j3AgGpf?usp=sharing>

- Falcon C++

Setelah melakukan instalasi IDE, kalian bisa menambahkan source file (halaman untuk menuliskan kode program) di pojok kiri atas. Pastikan supaya file tersebut menggunakan ekstensi **.c** (C File). Karena, untuk beberapa kode akan memberikan petunjuk pada IDE tentang jenis kode yang ada di dalam file tersebut.



Sesudah kalian membuat file dengan ekstensi **.c**, kalian bisa mulai menulis kode program di dalamnya. Disinilah kalian bisa mengimplementasikan algoritma, fungsi, variabel, dan berbagai elemen bahasa pemrograman C lainnya.

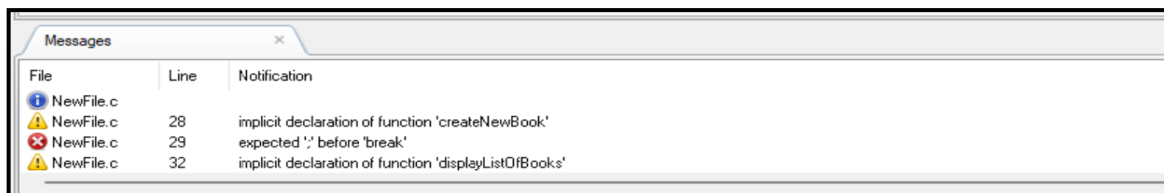


Untuk menuliskan kode program agar lebih praktis, kalian bisa menuliskan misal **main**, di Falcon sudah disediakan untuk template penulisannya, jadi seperti gambar di atas, kalian bisa memilih salah

satu sesuai kebutuhan kode program kemudian klik **tab** pada keyboard kalian, secara otomatis template kode program **main** akan tertulis di source file kalian. Tidak hanya main, Falcon juga menyediakan untuk kode/fungsi lainnya.



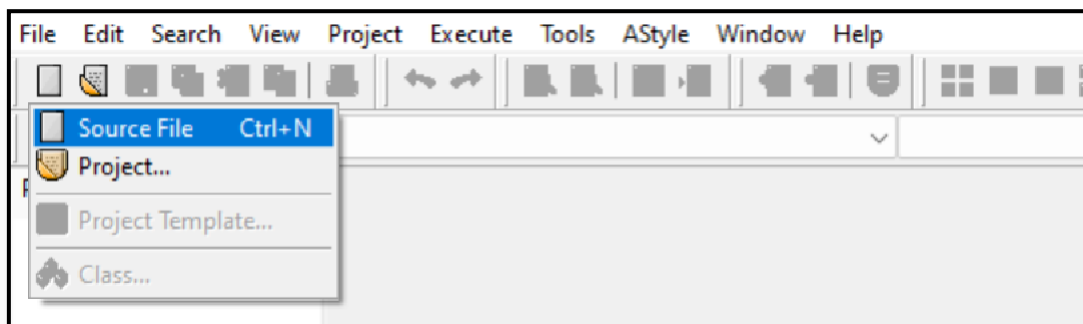
Untuk menjalankan program, kalian bisa klik tombol play di atas. Ketika program di jalankan, maka akan muncul terminal untuk menampilkan output dari kode program kalian. Jika tidak muncul, kalian bisa cek error yang ada dan perbaiki kode programnya.



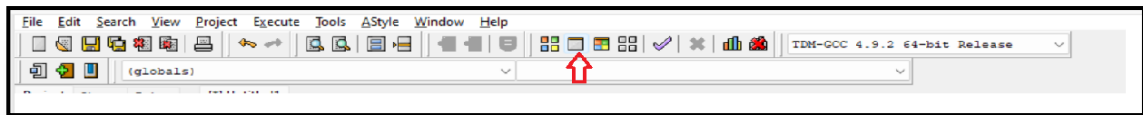
Untuk melihat pesan error kalian bisa cek kolom dibawah kode program kalian. Di kolom tersebut akan dijelaskan pada line berapa kode error dan alasan kenapa pada line tersebut mengakibatkan error. Jangan lupa untuk menyimpan file kode program kalian terlebih dahulu sebelum menjalankan program.

- Dev C++

Jika kalian menggunakan IDE Dev C++, kalian bisa membuat source file terlebih dahulu.



Setelah menambahkan source file, kalian bisa mulai untuk menulis kode program.



Untuk menjalankan kode program, kalian bisa klik tombol diatas atau gunakan f10.

### ➤ SYNTAX BAHASA C

```

1  int main()
2  {
3      //statement-statement
4  }
5
6  function()
7  {
8      //statement-statement
9  }
```

Struktur dari bahasa C dapat dilihat sebagai kumpulan dari sebuah atau lebih fungsi-fungsi. Fungsi pertama yang harus ada di program C sudah ditentukan namanya, yaitu **main()** yang kemudian dilanjutkan dengan kurung kurawal pembuka ({) dan diakhiri dengan kurung kurawal tutup (}). Diantara kurung kurawal ini, kalian bisa memasukkan statement-statement program. Dalam bahasa C, **main()** harus dituliskan karena merupakan fungsi utama agar program dapat berjalan. Selain itu, jangan pernah lupa untuk menambahkan **semicolon (;)** pada setiap akhir code baris. Apabila tertinggal satu saja, maka program akan *error* dan kalian harus semangat mencari letak *error* yang jarang kelihatan itu

Program dalam bahasa C terdiri dari beberapa komponen utama:

#### 1. Library Header

Library header adalah file yang berisi fungsi-fungsi yang dapat digunakan dalam program. Contohnya adalah library **stdio.h** yang menyediakan fungsi input/output seperti **printf** dan **scanf**. Library header dituliskan di awal program dengan menggunakan **#include**.

```
#include <stdio.h>
// standard input/output library
```

## 2. Fungsi Main

Setiap program C harus memiliki satu fungsi **main**. Fungsi ini menjadi titik mula eksekusi program. Di dalam fungsi **main**, kita menuliskan semua perintah yang ingin dijalankan oleh program.

```
int main() {  
    // Kode program ditulis di sini  
    return 0;  
}
```

## 3. Deklarasi Variables

Variabel harus dideklarasikan sebelum digunakan. Deklarasi variabel dilakukan dengan menyebutkan tipe data dan nama variabel.

```
int x; // variabel x dengan tipe data int  
float y; // variabel y dengan tipe data float  
char z; // variabel z dengan tipe data char
```

## 4. Statement

Statement adalah perintah atau instruksi yang dijalankan oleh program. Setiap statement diakhiri dengan tanda semicolon (;).

```
x = 10; // Statement untuk mengisi variabel x dengan  
        nilai 10  
printf("Nilai x: %d\n", x); // Statement untuk  
        mencetak nilai x ke layar
```



Berikut contoh sintaks pada bahasa C:

```
1.    #include <stdio.h>  
2.  
3.    main () {  
4.    printf("Hello World!");  
5.    return 0;  
6.    }
```

Penjelasan Code:

- Baris 1: **#include <stdio.h>** merupakan *preprocessor directive* yang menyertakan header file "**stdio.h**". Header ini berisi fungsi-fungsi standar input-output, termasuk "**printf**" yang digunakan dalam program ini.
- Baris 2: Baris kosong atau disebut juga *whitespace*. Dalam bahasa C, kita bisa menambahkan jarak antar baris tanpa memengaruhi output dari sebuah program. Biasanya digunakan supaya code program mudah dibaca.
- Baris 3: **main()** adalah deklarasi fungsi "**main**" yang merupakan titik masuk utama dalam program C. Program akan berjalan dari fungsi ini. Perlu diingat bahwa program tidak akan berjalan apabila tidak menggunakan fungsi **main()** alias akan menciptakan error.
- Baris 4: **printf("Hello World!")** adalah perintah untuk mencetak teks "**Hello World!**" ke layar. Fungsi **printf** digunakan untuk mencetak output ke layar.
- Baris 5: **return 0** adalah perintah untuk mengembalikan nilai 0 dari fungsi "**main**". Nilai 0 menandakan program berakhir dengan sukses. Jika program berakhir dengan nilai selain 0, itu menandakan ada masalah atau kesalahan.

## ➤ LIBRARY BAHASA C



```
#include <stdio.h>
#include <string.h>
#include <math.h>

int main()
{
    int age[5] = {16,45,67,14,27};
    printf("Before : %d", age[3]);

    age [3] = 23;
    printf("After : %d", age[3]);
    return 0;
}
```

Dalam pemrograman, library adalah kumpulan code yang telah dikompilasi sebelumnya dan menyediakan berbagai fungsi dan alat yang dapat digunakan oleh pengembang perangkat lunak (*software*). Library dapat berisi fungsi-fungsi untuk berbagai keperluan, seperti input/output, manipulasi string, pengolahan data, matematika dan masih banyak lagi. Kalian bisa menggunakan library ini dalam program dengan menyertakan header file yang sesuai dan mengandung deklarasi fungsi-fungsi dalam library. Penggunaan library tidak dibatasi, sesuai dengan program yang kalian butuhkan (bisa satu atau lebih). Ada dua jenis utama library dalam pemrograman yaitu:

- **Built-In-Library:** Merupakan library bawaan yang sudah tersedia secara default dalam bahasa pemrograman tertentu. Contohnya:
  1. **<stdio.h>** untuk fungsi-fungsi seperti **printf**, **scanf**, dan **getchar**
  2. **<stdlib.h>** untuk manajemen memori, konversi string ke angka, penggunaan fungsi **malloc** dan **free**, serta fungsi-fungsi utilitas lainnya.
  3. **<math.h>** untuk operasi matematika seperti **sqrt**, **sin**, **cos** dan sebagainya
  4. **<string.h>** untuk manipulasi string seperti **strlen**, **strcpy**, **strcmp**, dan lain-lain
  5. **<ctype.h>**: untuk operasi karakter seperti **isalpha**, **isdigit**, **toupper**, dan sejenisnya.
  6. **<time.h>**: untuk pengaturan waktu dan tanggal.

### Standard Input/Output Library

Header File: **stdio.h** Fungsi Utama: Menyediakan fungsi untuk input dan output standar, seperti **printf**, **scanf**, **getchar**, **putchar**, dll.

Contoh Penggunaan:

```
#include <stdio.h> // Meng-include library stdio.h
```

```
int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age); // Mengambil input dari pengguna

    printf("Your age is %d\n", age); // Menampilkan output ke layar

    return 0;
}
```

Penjelasan:

- **printf**: Fungsi untuk menampilkan output ke layar.
- **scanf**: Fungsi untuk mengambil input dari pengguna dengan format tertentu.

```
Enter your age: 12
Your age is 12
```

```
=== Code Execution Successful ===|
```

### Standard Library

Header File: `stdlib.h` Fungsi Utama: Menyediakan fungsi untuk alokasi memori, konversi angka, pengaturan proses, dan fungsi-fungsi utilitas lainnya.

Contoh Penggunaan:

```
#include <stdio.h>
#include <stdlib.h> // Meng-include library stdlib.h

int main() {
    char str[10] = "1234";
    int num;

    // Mengonversi string ke integer menggunakan fungsi atoi
    num = atoi(str);

    printf("String: %s, Integer: %d\n", str, num);

    return 0;
}
```

Penjelasan:

- **atoi**: Fungsi yang mengonversi string ke integer.

```
String: 1234, Integer: 1234
```

```
=== Code Execution Successful ===
```

### String Library

Header File: `string.h` Fungsi Utama: Menyediakan fungsi untuk memanipulasi string seperti copy, concatenate, compare, dan menghitung panjang string.

Contoh Penggunaan:

```
#include <stdio.h>
#include <string.h> // Meng-include library string.h

int main() {
    char str1[50] = "Pemrograman ";
    char str2[50] = "Bahasa C";

    strcat(str1, str2);

    printf("Hasil penggabungan: %s\n", str1);

    int len = strlen(str1);
    printf("Panjang string: %d\n", len);

    return 0;
}
```

Penjelasan:

- **strcat**: Fungsi untuk menggabungkan dua string.
- **strlen**: Fungsi untuk menghitung panjang string.

```
Hasil penggabungan: Pemrograman Bahasa C
Panjang string: 20
```

```
=== Code Execution Successful ===
```

### Mathematical Library

Header File: `math.h` Fungsi Utama: Menyediakan fungsi untuk operasi matematika seperti trigonometri, eksponensial, logaritma, dan akar kuadrat.

## Contoh Penggunaan:

```
#include <stdio.h>
#include <math.h> // Meng-include library math.h

int main() {
    double num = 16.0;
    double result;

    result = sqrt(num);

    printf("Akar kuadrat dari %.2f adalah %.2f\n", num, result);

    double angle = 45 * M_PI / 180;
    double sinValue = sin(angle);
    printf("Sinus dari 45 derajat adalah %.2f\n", sinValue);

    return 0;
}
```

## Penjelasan:

- **sqrt**: Fungsi untuk menghitung akar kuadrat.
- **sin**: Fungsi untuk menghitung nilai sinus dari sudut yang diberikan dalam radian.

```
Akar kuadrat dari 16.00 adalah 4.00
Sinus dari 45 derajat adalah 0.71

=== Code Execution Successful ===
```

- **External Library**: Merupakan library yang dikembangkan oleh pihak ketiga atau komunitas yang bisa diunduh dan digunakan dalam suatu program. Library ini memberikan fitur dan fungsi yang lebih khusus sesuai dengan kebutuhan tertentu. Untuk menggunakan library external, kalian perlu mengunduhnya dan menghubungkannya ke kode program kalian. Contoh external library di bahasa C adalah:

1. OpenGL: untuk grafika komputer 2D dan 3D.
2. SQLite: untuk manajemen database ringan.
3. SDL: untuk pengembangan permainan dan multimedia.
4. OpenSSL: untuk kriptografi dan keamanan.
5. Libcurl: untuk transfer data melalui berbagai protokol seperti HTTP dan FTP.
6. GNU Scientific Library (GSL): untuk komputasi ilmiah dan matematika.
7. Boost: Koleksi library yang memperluas fitur-fitur dalam bahasa C++, tetapi juga digunakan dalam bahasa C.

Penggunaan library selalu dicantumkan di awal code (baris pertama). Jika tidak menuliskan library yang diperlukan dalam codingan, maka program mungkin akan mengalami beberapa masalah tergantung pada library yang tidak disertakan. Kemungkinan yang terjadi diantaranya adalah:

1. Kompilasi Gagal: Saat proses kompilasi program, compiler akan menghasilkan kesalahan dan gagal menghasilkan file eksekusi.
2. Linking Error: Merupakan error yang akan terjadi apabila tidak menghubungkan program dengan library.
3. Runtime Error: Jika program berhasil dieksekusi dan di link, tetapi saat runtime program menggunakan fungsi yang seharusnya ada dalam library tertentu (tapi tidak ada), maka program akan menghasilkan error saat dijalankan dan mungkin akan crash.

Library dalam bahasa C sangat membantu dalam pengembangan program karena menyediakan fungsi-fungsi yang telah diuji dan siap digunakan. Menggunakan library dapat mempercepat proses pengembangan dan memastikan bahwa kode yang digunakan lebih efisien dan bebas dari bug. Pastikan untuk memahami fungsi-fungsi dalam library yang sering digunakan untuk meningkatkan produktivitas dalam pemrograman.

## TIPE DATA & VARIABEL

Tipe data dalam pemrograman berperan penting dalam mengatur bagaimana komputer akan menyimpan dan mengelola nilai-nilai yang ada dalam program. Setiap tipe data memiliki ukuran yang berbeda-beda, yang menentukan berapa banyak ruang memori yang dialokasikan untuk menyimpan nilai dari tipe data tersebut. Misalnya, tipe data **int** (integer) biasanya mengalokasikan 4 byte (32 bit) untuk menyimpan bilangan bulat, sedangkan tipe data **float** dan **double** digunakan untuk menyimpan bilangan pecahan dengan ukuran yang lebih besar.

Variabel, di sisi lain, adalah tempat penyimpanan yang diberi nama, yang digunakan untuk menyimpan nilai-nilai tertentu dalam program. Dalam deklarasi variabel, kita tidak hanya memberikan nama variabel, tetapi juga menentukan tipe data yang akan digunakan oleh variabel tersebut. Misalnya, dalam deklarasi **int nilai = 10;**, kita mendeklarasikan variabel **nilai** dengan tipe data **int** dan memberi nilai awal 10.

Ketika program dieksekusi, komputer mengalokasikan ruang memori yang cukup untuk menyimpan nilai dari variabel berdasarkan tipe datanya. Selanjutnya, kita bisa melakukan berbagai operasi pada variabel tersebut, seperti melakukan perhitungan matematika atau memanipulasi nilai. Misalnya, jika kita memiliki dua variabel **a** dan **b** dengan tipe data **int**, kita bisa melakukan operasi penjumlahan **a + b** atau pengurangan **a - b**.

Variabel dan tipe data bekerja bersama-sama dalam membentuk struktur dasar dari program. Pemahaman yang baik tentang tipe data dan variabel membantu pengembang dalam merancang program yang efisien, aman, dan mudah dimengerti. Selain itu, dengan memilih tipe data yang tepat untuk setiap variabel, kita dapat mengoptimalkan penggunaan memori dan meningkatkan performa program.

## TIPE DATA

Dalam bahasa C, terdapat beberapa tipe data diantaranya adalah **int**, **float**, **double** dan **char**. Masing-masing tipe data memiliki ciri khusus baik dari ukuran dan tujuan penggunaannya. Tipe data digunakan untuk memberikan tipe pada suatu variabel. Berikut adalah table dari jenis-jenis tipe data pada bahasa C:

TIPE DATA	SIZE	DESKRIPSI
<b>int</b>	2 atau 4 bytes	Untuk bilangan bulat

<b>float</b>	4 bytes	Untuk angka pecahan/desimal. Cukup untuk menyimpan 6-7 digit desimal
<b>double</b>	8 bytes	Untuk angka pecahan/desimal. Cukup untuk menyimpan 15 digit desimal
<b>char</b>	1 bytes	Untuk karakter/huruf/angka atau nilai ASCII

## Penggunaan Tipe Data dalam Program

### 1. Tipe data Integer

Tipe data integer dalam C digunakan untuk menyimpan bilangan bulat (bilangan apa pun termasuk positif, negatif, dan nol tanpa bagian desimal). Nilai oktal, nilai heksadesimal, dan nilai desimal dapat disimpan dalam tipe data int di C.

- Range: -2,147,483,648 to 2,147,483,647
- Size: 4 bytes
- Format Specifier: %d

#### Syntax Integer:

```
int var_name;
```

#### Contoh Implementasi Integer:

```
// C program to print Integer data types.
#include <stdio.h>

int main()
{
    // Nilai integer positif.
    int a = 9;

    // Nilai integer negatif.
    int b = -9;

    printf("Nilai integer a: %d\n", a);
    printf("Nilai integer b: %d\n", b);

    return 0;
}
```



Output:

```
Nilai Integer a: 9
Nilai Integer b: -9
```

## 2. Tipe data char

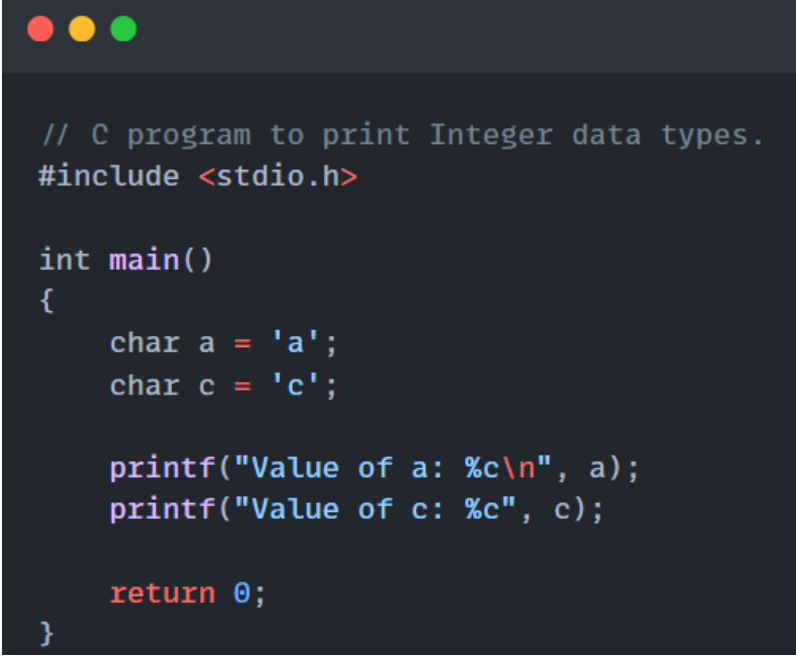
Tipe data char memungkinkan variabelnya untuk menyimpan hanya satu karakter. Ukuran karakter adalah 1 byte. Ini adalah tipe data paling dasar dalam C. Tipe ini menyimpan satu karakter dan membutuhkan satu byte memori di hampir semua kompiler.

- Range: (-128 to 127) or (0 to 255)
- Size: 1 byte
- Format Specifier: %c

Syntax Char:

```
char var_name;
```

Contoh Implementasi Char:



```
// C program to print Integer data types.
#include <stdio.h>

int main()
{
    char a = 'a';
    char c = 'c';

    printf("Value of a: %c\n", a);
    printf("Value of c: %c", c);

    return 0;
}
```

Output:

```
Value a: a
Value c: c
```

### 3. Tipe Data Float

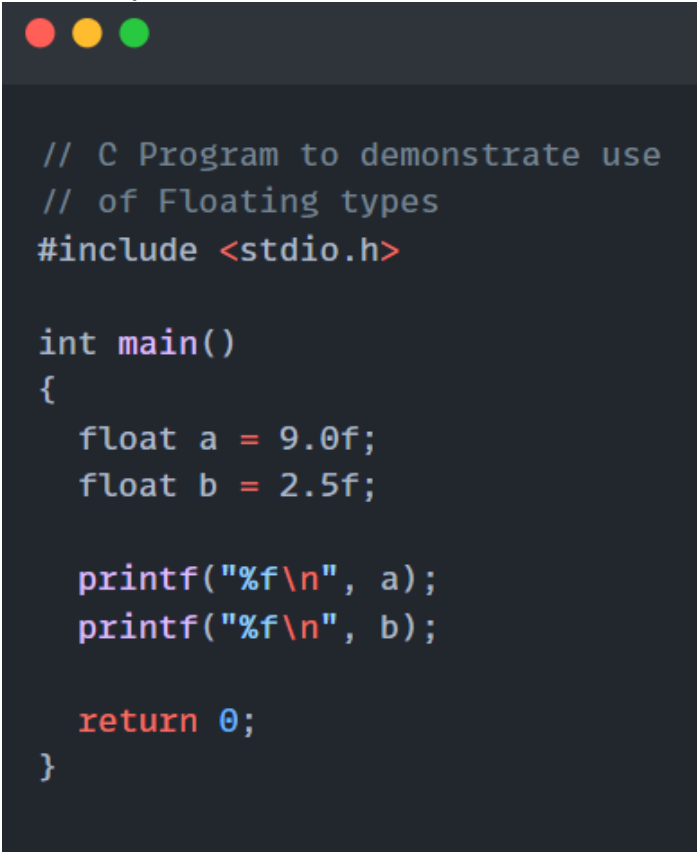
Dalam pemrograman C, tipe data float digunakan untuk menyimpan nilai floating-point. Float dalam C digunakan untuk menyimpan nilai desimal dan eksponensial. Tipe data ini digunakan untuk menyimpan angka desimal (angka dengan nilai floating point) dengan presisi tunggal.

- Range: 1.2E-38 hingga 3.4E+38
- Size: 4 byte
- Format Specifier: %f

Syntax Float:

```
float var_name;
```

Contoh Implementasi Float:

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is written in C and demonstrates the use of the float data type. It includes a comment, a header file, a main function, variable declarations, assignments, printf statements, and a return statement.

```
// C Program to demonstrate use
// of Floating types
#include <stdio.h>

int main()
{
    float a = 9.0f;
    float b = 2.5f;

    printf("%f\n", a);
    printf("%f\n", b);

    return 0;
}
```

Output:

```
9.000000
2.500000
```

### 4. Tipe Data Double

Tipe data Double dalam C digunakan untuk menyimpan angka desimal (angka dengan nilai floating point) dengan presisi ganda. Tipe data ini digunakan untuk mendefinisikan nilai numerik yang menyimpan angka dengan nilai desimal dalam C.

Tipe data double pada dasarnya adalah tipe data presisi yang mampu menyimpan 64 bit angka desimal atau floating point. Karena double memiliki presisi yang lebih tinggi dibandingkan dengan float, maka jauh lebih jelas bahwa tipe data ini menempati dua kali lipat memori yang digunakan oleh tipe floating-point. Tipe ini dapat dengan mudah menampung sekitar 16 hingga 17 digit setelah atau sebelum titik desimal.

- Range: 1.7E-308 to 1.7E+308
- Size: 8 bytes
- Format Specifier: %lf

#### Syntax Double:

```
double var_name;
```

#### Contoh Implementasi Double:

```
// C Program to demonstrate
// use of double data type
#include <stdio.h>

int main()
{
    double a = 123123123.00;
    double b = 12.293123;
    double c = 2312312312.123123;

    printf("%lf\n", a);

    printf("%lf\n", b);

    printf("%lf", c);

    return 0;
}
```

#### Output:

```
123123123.000000
12.293123
2312312312.123123
```

Untuk setiap tipe data memiliki format yang berbeda ketika akan melakukan input/output.

Berikut table untuk setiap format tipe data:

FORMAT	TIPE DATA
<b>%d</b> atau <b>%i</b>	int
<b>%f</b>	float
<b>%lf</b>	double
<b>%c</b>	char
<b>%s</b>	string

Untuk membuat program, pasti membutuhkan interaksi dengan user berupa input/output. Maka dari itu, dibutuhkan format-format khusus yang disediakan oleh fungsi-fungsi bawaan, seperti `scanf` dan `printf`. Pada saat kita meminta input dari pengguna dengan menggunakan fungsi **`scanf`**, kita menggunakan format-format seperti **`%d`** untuk menerima input berupa bilangan bulat (integer) dan **`%c`** untuk menerima input berupa karakter. Misalnya, jika kita ingin mengambil input berupa umur dari user, kita dapat menggunakan **`%d`** dalam fungsi **`scanf`**.

```
int age;
scanf("%d",
```

Format **`%d`** akan memberi tahu **`scanf`** bahwa input yang dibutuhkan adalah bilangan bulat. Kemudian, nilai yang dimasukkan oleh user akan disimpan dalam variabel **`age`**.

Sementara itu, untuk menampilkan output menggunakan fungsi **printf**, kita bisa menggunakan format-format yang serupa untuk memformat data yang akan ditampilkan. Misalnya kita akan menggunakan **%d** untuk menampilkan nilai dari variabel integer.

```
int score = 85;
printf("Your score is: %d\n", score);
```

Dari contoh di atas, **%d** akan digunakan untuk memberi tahu **printf** bahwa kita ingin menampilkan nilai integer dari variabel **'score'**. Jadi, penggunaan format ini penting untuk memastikan bahwa input yang dimasukkan sesuai dengan tipe data yang dibutuhkan dan output yang dihasilkan terformat dengan baik.

Contoh penggunaan format dalam type data

```
#include <stdio.h>

int main() {
    int usia;
    float tinggi;
    char inisial;

    printf("Masukkan usia: ");
    scanf("%d", &usia);
    printf("Masukkan tinggi (cm): ");
    scanf("%f", &tinggi);
    printf("Masukkan inisial nama: ");
    scanf(" %c", &inisial);
    printf("Usia: %d, Tinggi: %.2f cm, Inisial: %c\n", usia,
tinggi, inisial);

    return 0;
}
```

Penjelasan:

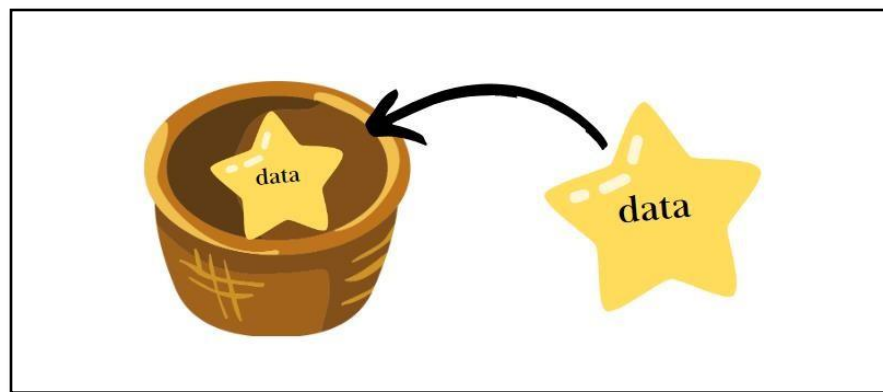
- Baris 7-8: Menggunakan scanf untuk mengambil input dari pengguna dengan spesifikasi format yang sesuai (%d, %f, %c).
- Baris 15: Menampilkan hasil input menggunakan printf.

```
Masukkan usia: 12
Masukkan tinggi (cm): 170
Masukkan inisial nama: a
Usia: 12, Tinggi: 170.00 cm, Inisial: a

=== Code Execution Successful ===
```

## VARIABEL

Variabel adalah suatu entitas yang digunakan untuk menyimpan nilai atau data dalam suatu program komputer. Dalam bahasa pemrograman, variabel memiliki tipe data tertentu yang menentukan jenis nilai yang dapat disimpan di dalamnya, seperti angka, karakter, atau nilai . Setiap variabel memiliki nama yang digunakan untuk mengidentifikasinya dalam program. Nama variabel harus mengikuti aturan penamaan yang berlaku dalam bahasa pemrograman tertentu dan nama tersebut tidak boleh mengandung spasi atau karakter khusus tertentu. Selain itu, nama variabel harus unik dalam ruang lingkup (scope) program, artinya tidak dapat ada dua variabel dengan nama yang sama dalam lingkup yang sama.



Setiap variabel memiliki beberapa atribut penting yang harus diperhatikan:

1. **Nama Variabel:** Nama variabel adalah identifikasi unik yang digunakan untuk merujuk pada nilai yang disimpan. Nama variabel harus diikuti oleh aturan penamaan yang berlaku dalam bahasa pemrograman tertentu. Nama variabel ini juga haruslah deskriptif sehingga memudahkan programmer lain untuk memahami tujuan variabel tersebut.
2. **Tipe Data:** Tipe data variabel menentukan jenis nilai yang dapat disimpan di dalamnya. Bahasa pemrograman umumnya memiliki tipe data primitif seperti angka bulat (integer), desimal (float/double), karakter (char), dan boolean (true/false). Tipe data mempengaruhi bagaimana variabel akan diperlakukan dan dioperasikan dalam program.
3. **Nilai:** Variabel menyimpan nilai atau data yang sesuai dengan tipe datanya. Nilai ini bisa berupa angka, teks, atau data lainnya yang relevan dengan konteks program.
4. **Scope:** Scope adalah ruang lingkup dimana variabel dideklarasikan dan dapat digunakan. Variabel hanya dapat diakses dan digunakan dalam ruang lingkup tertentu, seperti dalam blok kode tertentu atau dalam fungsi tertentu.

## ATURAN PENAMAAN VARIABEL

Aturan umum penamaan variabel dalam bahasa C adalah:

- 1) Nama variabel harus dimulai dengan huruf (a-z, A-Z) atau karakter underscore (\_)
- 2) Setelah karakter pertama, nama variabel dapat mengandung huruf, angka (0-9), atau karakter underscore (\_).
- 3) Nama variabel tidak boleh diawali dengan angka. Contoh : **'123name'**, **'2count'**.
- 4) Nama variabel bersifat case-sensitive, artinya huruf besar dan huruf kecil dianggap berbeda. Misalnya, angka dan Angka adalah dua variabel yang berbeda.
- 5) Nama variabel tidak boleh mengandung spasi atau karakter khusus seperti !, @, #, \$, %, ^, &, \*, (, ), dan sebagainya.
- 6) Variabel tidak boleh menggunakan kata kunci bawaan dalam bahasa C sebagai nama. Contohnya, kalian tidak dapat menggunakan **int**, **float**, **for**, dan sejenisnya sebagai nama variabel karena sudah memiliki arti khusus dalam bahasa C.
- 7) Pendekatan *camel case* umum digunakan dalam penamaan variabel. Huruf pertama dari kata pertama menggunakan huruf kecil dan huruf pertama dari kata-kata berikutnya menggunakan huruf besar. Contoh : **'totalAmount'**, **'numberOfStudents'** dan sebagainya.
- 8) Pilihlah nama variabel yang deskriptif agar mudah dimengerti oleh orang lain yang membaca kode kalian. Hindari nama variabel yang terlalu singkat dan ambigu, seperti **'x'**, **'y'**, **'temp'** kecuali jika konteksnya sangat jelas.

Tujuan dari aturan penamaan variabel adalah supaya kode program mudah dibaca dan dipahami baik untuk kalian sendiri maupun orang lain yang mungkin akan membaca atau mengembangkan kode tersebut.

### Contoh penamaan variabel yang benar:

```
int age;  
float salary;  
char _status;  
double average_score;  
int studentCount;
```

Contoh penamaan variabel yang salah:

```
int 1stNumber; // Starts with a digit
float my-salary; // Contains a hyphen (-)
char int; // Same as a C keyword
int double; // Same as a C keyword
float my$var; // Contains an unsupported special character
```

## JENIS-JENIS VARIABEL DALAM C

Ada banyak jenis variabel dalam bahasa C:

1. variabel lokal
2. variabel global
3. variabel statis
4. variabel otomatis
5. variabel eksternal

### Variabel Lokal

Variabel yang dideklarasikan di dalam fungsi atau blok disebut variabel lokal. Variabel ini harus dideklarasikan di awal blok.

```
void function1(){
    int x=10;//local variable
}
```

Kita harus menginisialisasi variabel lokal sebelum digunakan.

### Variabel Global

Variabel yang dideklarasikan di luar fungsi atau blok disebut variabel global. Setiap fungsi dapat mengubah nilai variabel global. Ini tersedia untuk semua fungsi. Variabel ini harus dideklarasikan pada awal blok.



```
int value=20;//global variable
void function1(){
int x=10;//local variable
}
```

### Variabel Statis

Variabel yang dideklarasikan dengan kata kunci `static` disebut variabel statis. Variabel ini mempertahankan nilainya di antara beberapa pemanggilan fungsi.

```
void function1(){
int x=10;//local variable
static int y=10;//static variable
x=x+1;
y=y+1;
printf("%d,%d",x,y);
}
```

Jika kita memanggil fungsi ini berkali-kali, variabel lokal akan mencetak nilai yang sama untuk setiap pemanggilan fungsi, misal, 11,11,11 dan seterusnya. Tetapi variabel statis akan mencetak nilai yang bertambah di setiap pemanggilan fungsi, misalnya 11, 12, 13, dan seterusnya.

### Variabel Otomatis

Semua variabel dalam bahasa C yang dideklarasikan di dalam blok, adalah variabel otomatis secara default. Kita dapat mendeklarasikan variabel otomatis secara eksplisit dengan menggunakan *auto keyword*.

```
void main(){
int x=10;//local variable (also automatic)
auto int y=20;//automatic variable
}
```

## Variabel Eksternal

Kita dapat berbagi sebuah variabel dalam beberapa file sumber C dengan menggunakan variabel eksternal. Untuk mendeklarasikan variabel eksternal, Anda perlu menggunakan kata kunci `extern`.

*Myfile.h*

```
extern int x=10; //external variable (also global)
```

*Program1.c*

```
#include "myfile.h"  
#include <stdio.h>  
void printValue(){  
    printf("Global variable: %d", global_variable);  
}
```

## DEKLARASI VARIABEL

Dalam bahasa C, kalian bisa mendeklarasikan variabel dengan mengikuti sintaks berikut :

```
TipeData NamaVariabel = NilaiVariabel
```

Penjelasan sintaks:

- TipeData: Tipe data dari variabel yang ingin kalian deklarasikan seperti **int**, **float**, **char**, **double**, **bool**, dll.
- NamaVariabel: Nama yang ingin kalian berikan untuk variabel tersebut. Ingat, penamaan harus sesuai aturan penamaan variabel yang sudah dijelaskan sebelumnya.
- NilaiVariabel: Apabila kalian ingin variabel sudah memiliki nilai ketetapan, maka kalian bisa langsung mendeklarasikannya, contoh:

```
int number = 23;
char my_name = 'Jaemin'
```

Namun, jika kalian tidak memerlukan nilai dari variabel tersebut (hanya membuat variabel), maka tidak perlu menuliskan nilai variabel.

Jika kalian ingin mendeklarasikan lebih dari satu variabel namun memiliki tipe data yang sama, kalian bisa membuatnya menjadi **multiple variables** yaitu:

```
contoh 1 => int x = 1, y = 2, z = 3;

contoh 2 => int x, y, z;
             x=y=z= 120;
```

Berikut, kalian bisa copy paste ke IDE kalian untuk mengetahui cara membuat variabel dalam bahasa C

```
#include <stdio.h>

int main () {
    int a = 10; //membuat variabel a dengan tipe data int
    int b = 20; //membuat variabel b dengan tipe data int
    int sum = a + b; //membuat variabel sum dengan tipe data int

    printf("Hasil a + b = %d", sum);
}
```

## INPUT & OUTPUT

Ketika kita memasuki suatu aplikasi/website, pasti kita diarahkan untuk Login/Signup sehingga membutuhkan input dari user. Maka dari itu, saat membuat program kalian pasti akan membutuhkan input dari user dan output yang akan ditampilkan di layar. Di bahasa C, terdapat beberapa fungsi yang bisa kita gunakan untuk input/output.

Untuk menambahkan baris baru bisa menggunakan “\n” dan untuk memberikan tab kita bisa menggunakan “\t”.

Contoh penggunaan \n, bisa kalian copy-paste kode program di bawah ini

```
#include <stdio.h>

int main() {
    printf("Halo,\nSelamat datang\ndi dunia\npemrograman!");return 0;
}
```

Dari kode di atas, setelah “halo” akan menambahkan baris baru. Sehingga, kalimat “selamat datang” akan berada di baris baru, bukan di baris sebelumnya. Begitu juga dengan kalimat “di dunia pemrograman” akan berada di baris baru. Sehingga, dari kode program di atas akan berada di 3 baris yang berbeda.

Untuk penggunaan `\t`, kalian bisa copy-paste kode program di bawah ini

```
#include <stdio.h>

int main() {
    printf("Nama\t: John\nUmur\t: 25\nKota\t:
    Jakarta");return 0;
}
```

Pada program di atas, `\t` digunakan untuk memasukkan karakter tab (indentasi horizontal) dalam teks. Ini membantu memberikan jarak yang konsisten antara kata-kata seperti "Nama:", "Umur:", dan "Kota:" sehingga membuat tampilan lebih rapi. Sementara itu, `\n` digunakan untuk memindahkan cursor ke baris baru setelah setiap kalimat, sehingga setiap kalimat ("Nama: John", "Umur: 25", dan "Kota: Jakarta") akan tampil pada baris yang berbeda, memberikan tampilan yang lebih terstruktur dan mudah dibaca.

Selanjutnya, berikut contoh fungsi input yang bisa kalian gunakan:

- **scanf()** digunakan untuk mengambil input dari pengguna dengan menggunakan format yang ditentukan.

Format ini menandakan tipe data yang akan diambil sebagai input.

**Sintaks:**

```
int scanf(const char *format, ...);
```

**Spesifikasi Format:**

Format Specifier	Tipe Data	Penjelasan
<b>%d</b>	int	Bilangan bulat (decimal)
<b>%f</b>	float	Bilangan pecahan (decimal)
<b>%lf</b>	double	Bilangan pecahan (presisi ganda)
<b>%c</b>	char	Karakter tunggal
<b>%s</b>	char[] (string)	String (menghentikan input pada whitespace)
<b>%u</b>	unsigned int	Bilangan bulat tanpa tanda

**Contoh Penggunaan scanf:**

```
#include <stdio.h>

int main() {
    int umur;
    float tinggi;
    char gender;
```

```
printf("Masukkan umur: ");
scanf("%d", &umur);

printf("Masukkan tinggi badan (cm): ");
scanf("%f", &tinggi);

printf("Masukkan jenis kelamin (L/P): ");
scanf(" %c", &gender);

printf("Umur: %d, Tinggi: %.2f cm, Jenis Kelamin: %c\n", umur,
tinggi, gender);

return 0;
}
```

Penjelasan:

- Baris 9-10: `scanf("%d", &umur);` mengambil input integer untuk variabel **umur**.
- Baris 12-13: `scanf("%f", &tinggi);` mengambil input float untuk variabel **tinggi**.
- Baris 15-16: `scanf(" %c", &gender);` mengambil input karakter untuk variabel **gender**.

Ouput:

```
Masukkan umur: 20
Masukkan tinggi badan (cm): 175
Masukkan jenis kelamin (L/P): L
Umur: 20, Tinggi: 175.00 cm, Jenis Kelamin: L

=== Code Execution Successful ===
```

- **gets()**: Fungsi ini digunakan untuk mengambil input dalam satu baris dan tidak memerlukan format seperti **scanf()**.

Sintaks:

```
char *gets(char *str);
```

Contoh Penggunaan:

```
#include <stdio.h>

int main() {
    char name[50];
    printf("Masukkan nama Anda: ");
    gets(name);
    printf("Halo, %s!", name);
    return 0;
}
```

```
Masukkan nama Anda: aditya
Halo, aditya!
```

```
=== Code Execution Successful ===
```

Perhatikan kode di atas. Berbeda dengan fungsi **scanf()**, fungsi **gets()** hanya membutuhkan variabel **name** didalamnya (tanpa menggunakan format). Jadi, inputan dari user akan otomatis tersimpan ke dalam variabel **name** dan kemudian akan dipanggil pada fungsi **printf()** menggunakan format **%s**. Kenapa **%s** alias string? Karena kode di atas menggunakan tipe data **char** yang menggunakan array. Supaya lebih jelasnya, nanti akan kalian pelajari di modul selanjutnya. Jadi untuk sekarang, pahami dulu basic dari input/output pada pemrograman :)

- **fgets()**: Fungsi ini bisa menentukan ukuran buffer dan sumber inputan. Ukuran buffer merupakan batas ukuran panjang string yang diinputkan, sedangkan "**stdin**" adalah sumber inputan dari keyboard.

```
#include <stdio.h>
#include <string.h>

int main() {
    char name[50];
    printf("Masukkan nama Anda: ");
    fgets(name, sizeof(name), stdin);

    name[strcspn(name, "\n")] = '\0';

    printf("Halo, %s!", name);
    return 0;
}
```

```
Masukkan nama Anda: aditya
Halo, aditya!
```

```
=== Code Execution Successful ===
```

Perhatikan kode di atas. Fungsi **fgets** digunakan untuk membaca input string dari pengguna dan memasukkannya ke dalam array **name**. **sizeof(name)** digunakan untuk membatasi ukuran buffer agar tidak terjadi **buffer overflow**. Buffer overflow adalah sebuah keadaan di mana data yang melebihi kapasitas dari suatu buffer (sebuah arena penyimpanan sederhana dalam memori) tertulis ke dalam buffer tersebut, dan akhirnya dapat menimpa data yang ada di lokasi memori yang seharusnya tidak berpengaruh.



Sedangkan untuk fungsi output, kalian bisa menggunakan fungsi berikut :

- **printf():** Fungsi ini terdapat di library **stdio.h**. Fungsi ini digunakan untuk menampilkan output ke layar dengan menggunakan format yang ditentukan.

**Sintaks:**

```
int printf(const char *format, ...);
```

**Spesifikasi Format:**

Format Specifier	Tipe Data	Penjelasan
%d	int	Bilangan bulat (decimal)
%f	float	Bilangan pecahan (decimal)
%lf	double	Bilangan pecahan (presisi ganda)
%c	char	Karakter tunggal
%s	char[] (string)	String (menghentikan input pada whitespace)
%u	unsigned int	Bilangan bulat tanpa tanda

**Contoh Penggunaan:**

```
#include <stdio.h>

int main() {
    int num1 = 10, num2 = 20;
    printf("Angka pertama: %d, Angka kedua: %d", num1, num2);
    return 0;
}
```

```
Angka pertama: 10, Angka kedua: 20
```

```
=== Code Execution Successful ===
```

Dari kode di atas, **printf** akan mencetak output ke layar. Dengan menambahkan format tanda petik (" ") di dalam tanda kurung akan memunculkan output sesuai dengan format yang dituliskan.

- **puts()**: Fungsi ini sama seperti **printf()**, bedanya adalah tidak perlu menggunakan format dan akan selalu membuat baris baru tanpa harus menggunakan symbol “\n”.

Sintaks:

```
int puts(const char *str);
```



```
#include <stdio.h>

int main() {
    char message[] = "Hello World!";
    puts(message);
    return 0;
}
```

```
Hello World!
```

```
=== Code Execution Successful ===
```

Dari kode di atas, fungsi **puts** hanya menampilkan dari variabel yang sudah diinisialisasikan nilainya. Jadi, tidak perlu menggunakan format seperti **printf**, fungsi **puts** akan langsung menampilkan output yang dibutuhkan

Coba untuk menulis ulang kode dibawah ini agar kalian memahami konsep input/output pada pemrograman.

```
#include
<stdio.h>int
main() {
int nilai;
printf("Masukkan Nilai Anda:
");scanf("%d", &nilai);
printf("Nilai Anda Adalah: ", &nilai);
```

Dari contoh di atas, "%d" berarti kalian ingin menginputkan data berupa bilangan bulat (integer) dari pengguna, dan operator "&nilai" digunakan untuk mendapatkan alamat memori variabel nilai, sehingga data yang diinput oleh pengguna akan disimpan pada alamat memori variabel nilai.

## COMMENTS

Jika ingin menambahkan penjelasan pada code, kita bisa menambahkan komentar. Hal ini juga dapat mencegah eksekusi ketika menguji code alternatif. Comments tidak akan dieksekusi oleh compiler kecuali jika salah sintaks maka akan menimbulkan error.

Kapan dan Mengapa menggunakan Komentar dalam pemrograman C?

1. Seseorang yang membaca kode yang besar akan kebingungan jika tidak ada komentar yang diberikan tentang detail program.
2. Komentar C adalah cara untuk membuat kode lebih mudah dibaca dengan memberikan lebih banyak deskripsi.
3. Komentar C dapat menyertakan deskripsi algoritma untuk membuat kode dapat dimengerti.
4. C Komentar dapat digunakan untuk mencegah eksekusi beberapa bagian kode.



- **Single-line comments**


Single line comment digunakan untuk memberikan komentar dalam satu baris kode.

Komentar ini tidak akan dijalankan oleh kompiler dan hanya digunakan untuk memberikan penjelasan atau dokumentasi pada kode. Single line comment dimulai dengan tanda `//` dan berlaku hingga akhir baris.

Syntax single line comment:

```
// This is a single line comment
```

Contoh single line comment:



```
#include <stdio.h>

//membuat fungsi penilaian
int main() {
    char grade = 'A'; //menggunakan tipe data char
    printf("Nilai Anda: %c\n", grade);
    return 0;
}
```

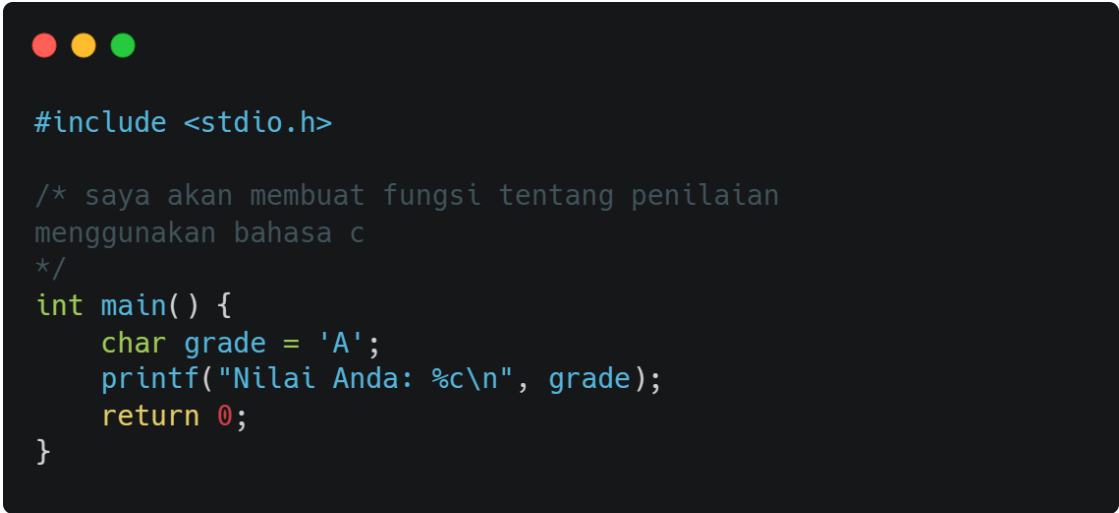
- **Multi-line comments**

Multi line comment, juga dikenal sebagai block comment, digunakan untuk memberikan komentar pada beberapa baris kode. Komentar ini biasanya digunakan untuk memberikan penjelasan yang lebih detail atau untuk menonaktifkan sejumlah baris kode sementara waktu. Multi line comment dimulai dengan `/*` dan diakhiri dengan `*/`.

Syntax multi line comment:

```
/*Comment starts
   continues
   continues
   .
   .
   .
Comment ends*/
```

Contoh implementasi multi line comment:



```
#include <stdio.h>

/* saya akan membuat fungsi tentang penilaian
menggunakan bahasa c
*/
int main() {
    char grade = 'A';
    printf("Nilai Anda: %c\n", grade);
    return 0;
}
```

Biasanya penggunaan comment dilakukan untuk memberikan keterangan penjelasan pada suatu kode agar developer lain bisa lebih mudah untuk memahami alur kode program. Selain mempermudah developer lain untuk memahami alur kode, tetapi juga bisa membantu dalam berbagai aspek pengembangan perangkat lunak yang melibatkan kolaborasi, pemeliharaan, dan perbaikan.

Dalam lingkungan pengembangan tim atau *open-source*, komentar yang baik menjadi jembatan komunikasi antara pengembang yang mungkin memiliki latar belakang, pengetahuan, atau keahlian yang berbeda. Misalnya, komentar bisa memberikan penjelasan tentang algoritma yang kompleks, pemilihan struktur data tertentu, atau keputusan desain yang diambil dalam kode.

Komentar juga bisa membantu kita untuk mengingat kembali alasan kenapa kita menuliskan program tersebut. Kode yang ditulis hari ini mungkin akan terlupaka, jadi dengan menambahkan komentar kita bisa mengingatkannya kembali. Tapi, kalau terlalu banyak komentar juga akan membuat kode menjadi kacau dan sulit dipahami. Jadi, lebih baik berikan komentar yang kompleks dan *to the point*.

## CODELAB 1

Tulis ulang kode program di bawah ini ke IDE kalian. Cobalah untuk memperbaikinya, jika sudah tunjukkan ke asisten masing - masing dan jelaskan apa saja yang telah kalian perbaiki.

```
#include <stdio.h>
int main()
{
    int a, b;
    printf("Masukkan dua angka: ");
    scanf("%d %d", a, b);

    int sum = a + b;
    printf("Penjumlahan: %d\n", sum);

    int difference = a - b;
    printf("Pengurangan: %d\n", difference);

    int product = a * b;
    printf("Perkalian: %d\n", product);

    if (b != 0) {
        float division = a / b;
        printf("Pembagian: %.2f\n", division);
    } else {
        printf("Error: Pembagian dengan nol\n");
    }

    return 0;
}
```

**CODELAB 2**

Tulis ulang kode program di bawah ini ke IDE kalian. Cobalah untuk memperbaikinya, jika sudah tunjukkan ke asisten masing - masing dan jelaskan apa saja yang telah kalian perbaiki

```
#include <stdio.h>
int main()
{
    char nama[50];
    int umur;
    float gaji_pokok;
    printf("Masukkan nama: ");
    gets(nama);

    printf("Masukkan umur: ");
    scanf("%d", umur);

    printf("Masukkan gaji pokok: ");
    scanf("%f", gaji_pokok);

    float tunjangan = gaji_pokok * 0.1;
    float total_gaji = gaji_pokok +
    tunjangan;
    printf("Nama: %s\n", nama)
    printf("Umur: %d\n", umur);
    printf("Gaji Pokok: %.2f\n", gaji_pokok);
    printf("Tunjangan: %.2f\n", tunjangan);
    printf("Total Gaji: %.2f\n", total_gaji);
}
```

Setelah mengerjakan codelab, tunjukkan ke asisten masing-masing untuk penilaian.



## KEGIATAN 1

### Pembuatan Program Penghitung Nilai Mahasiswa

Instruksi: Buatlah sebuah program untuk menghitung nilai akhir mahasiswa di mana program ini harus mencakup:

1. Pengenalan Program (menggunakan fungsi printf).
2. Mengambil input dari pengguna untuk:
  - a. Nama Mahasiswa
  - b. NIM Mahasiswa
  - c. Nilai Tugas (dalam persentase 20% dari nilai akhir)
  - d. Nilai UTS (dalam persentase 35% dari nilai akhir)
  - e. Nilai UAS (dalam persentase 45% dari nilai akhir)
3. Menghitung nilai akhir berdasarkan persentase yang telah diberikan.
4. Menampilkan hasil perhitungan nilai akhir beserta informasi mahasiswa.

```
--- Program Penghitung Nilai Akhir Mahasiswa ---  
Masukkan Nama Mahasiswa: aditya  
Masukkan NIM Mahasiswa: 0000  
Masukkan Nilai Tugas (0-100): 80  
Masukkan Nilai UTS (0-100): 90  
Masukkan Nilai UAS (0-100): 95  
-----  
Nama Mahasiswa: aditya  
NIM Mahasiswa: 0000  
Nilai Tugas: 80.00  
Nilai UTS: 90.00  
Nilai UAS: 95.00  
Nilai Akhir: 90.25  
-----  
=== Code Execution Successful ===
```

---

**KRITERIA & DETAIL PENILAIAN**

Kriteria	Poin
Codelab 1	10
Codelab 2	10
Kegiatan 1	30
Pemahaman	30
Ketepatan Menjawab	20