

# Konsep Pemrograman

## 2. Dasar-dasar Pemrograman C

Umi Sa'adah

Entin Martiana Kusumaningtyas

Tri Hadiah Muliawati

2020



Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika dan Komputer

# Overview

- Tipe Data Standar (*Standart Data Type*)
- Aturan Pendefinisian Identifier
- Variabel
  - Deklarasi
  - Inisialisasi
- Konstanta
- Operator
  - Operator Aritmatika
  - Operator Penurunan dan Peningkatan
  - Prioritas Operator Aritmatika
  - Operator Penugasan
  - Operator Kombinasi (Pemendekan)
- Operasi I/O



# Tipe Data Standar

- Berdasarkan jenis/tipenya, data dapat dibagi menjadi lima kelompok, yang dinamakan sebagai tipe data dasar.
  - Bilangan bulat (integer) □ int (short int, long int, signed int, unsigned int)
  - Bilangan real presisi-tunggal □ float
  - Bilangan real presisi-ganda □ double
  - Karakter □ char



# Ukuran Memori Tipe Data

Tipe	Total bit	Kawasan	Keterangan
char	8	-128 s/d 127	karakter
int	32	-2147483648 s/d 2147483647	bilangan integer
float	32	1.7E-38 s/d 3.4E+38	bilangan real presisi-tunggal
double	64	2.2E-308 s/d 1.7E+308	bilangan real presisi-ganda

Tipe	Total bit	Kawasan	Keterangan
short int	16	-32768 s/d 32767	short integer
long int	32	-2147483648 s/d 2147483647	long integer
signed int	32	-2147483648 s/d 2147483647	biasa disingkat dengan int
unsigned int	32	0 s/d 4294967295	bilangan int tak bertanda

# Aturan Penamaan Identifier

(variable, konstanta, dan fungsi)

- Identifier harus diawali dengan huruf (A..Z, a..z) atau karakter garis bawah ( \_ ).
- Selanjutnya dapat berupa huruf, digit (0..9) atau karakter garis bawah atau tanda dollar (\$).
- Panjang pengenalan boleh lebih dari 31 karakter, tetapi hanya 31 karakter pertama yang akan dianggap berarti.
- Pengenalan tidak boleh menggunakan nama yang tergolong sebagai kata-kata cadangan (reserved words) seperti int, if, while dan sebagainya

# Variabel vs Konstanta

- Data merupakan suatu nilai yang bisa dinyatakan dalam bentuk konstanta atau variabel.
  - Konstanta menyatakan nilai yang tetap.
  - Variabel menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung.

# Variabel: Deklarasi

- Variabel yang akan digunakan dalam program harus dideklarasikan terlebih dahulu.
  - Deklarasi berarti program memesan memori dan menentukan jenis/tipe data yang bisa disimpan di dalamnya.
  - Bentuk umum deklarasi variabel:
- ```
tipe-data nama-variabel;
```
- Pada pendeklarasian variabel, daftar-variabel dapat berupa sebuah variabel atau beberapa variabel yang dipisahkan dengan koma. Contoh:

```
int bil;  
float luas, radius;
```

# Variabel: Inisialisasi

- Inisialisasi berarti pemberian nilai awal pada variabel.
- Adakalanya variabel langsung diberi nilai awal setelah dideklarasikan, misal:

```
int bil, total;  
bil = 10;  
total = 0;
```

- Dua pernyataan di atas sebenarnya dapat disingkat melalui pendeklarasian yang disertai penugasan nilai, sebagai berikut :

```
int bil = 10, total=0;
```

- Penulisan di atas dapat menghemat penggunaan baris kode (*line of code*), selain itu penulisan di atas juga dapat memberikan kejelasan, khususnya untuk variabel yang perlu diberi nilai awal (diinisialisasi).





# Konstanta

- Pendefinisian konstanta dapat dilakukan menggunakan 2 cara:

1. Menggunakan preprocessor directive `#define` dengan tanpa diakhiri dengan titik koma

```
#define HRF 'A'  
#define PI 3.14f  
#define MAX 10  
#define DEC 27.5  
#define KALIMAT "Deklarasi Konstanta"
```

2. Menggunakan *keyword* `const`

```
const char HRF = 'A';  
const float PI = 3.14f;  
const int MAX = 10;  
const double DEC = 27.5;  
const char KALIMAT[] = "Deklarasi Konstanta";
```

- Penulisan konstanta mempunyai aturan tersendiri, sesuai dengan tipe data yang digunakan.



# Operator

- Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seperti menjumlahkan dua buah nilai, memberikan nilai ke suatu variabel, membandingkan kesamaan dua buah nilai.

- Berdasarkan jumlah *operand*-nya :

1. Unary operator, contoh : `-c`

Operator yang hanya memiliki sebuah *operand*. *Operand* pada contoh di atas adalah `c`.

2. Binary operator, contoh : `a + b`

Operator yang dikenakan terhadap dua buah *operand*. *Operand* pada contoh di atas adalah `a` dan `b`.

3. Ternary operator, contoh : `?` :

```
hasil = (x > y) ? 0 : 1;
```

Arti dari *statement* di atas adalah sbb:

Apakah nilai variabel `x` lebih besar dari nilai variabel `y`? jika `true`, maka nilai `0` akan dimasukkan ke dalam variabel `hasil`. Jika `false`, maka nilai `1` akan dimasukkan ke dalam variabel `hasil`.



# Operator Aritmatika

| Operator | Operasi                | Kategori | Contoh       |
|----------|------------------------|----------|--------------|
| *        | Perkalian              | Binary   | $2 * 3 = 6$  |
| /        | Pembagian / Hasil bagi | Binary   | $4 / 2 = 2$  |
| +        | Penjumlahan            | Binary   | $4 + 9 = 13$ |
| -        | Pengurangan            | Binary   | $4 - 5 = -1$ |
| %        | Modulus / Sisa bagi    | Binary   | $4 \% 2 = 0$ |
| +        | Tanda plus             | Unary    | $+8 = 8$     |
| -        | Tanda minus            | Unary    | $-8 = -8$    |

# Prioritas Operator Aritmatika

| PRIORITAS | OPERATOR                        | URUTAN Pengerjaan     |
|-----------|---------------------------------|-----------------------|
| Tertinggi | ( )                             | dari kiri ke kanan    |
|           | !    ++    --    +    -         | dari kanan ke kiri *) |
|           | *    /    %                     | dari kiri ke kanan    |
|           | +    -                          | dari kiri ke kanan *) |
| Terendah  | =    +=    -=    *=    /=    %= | dari kanan ke kiri    |

# Operator Penjumlahan dan Perkalian

```
1 #include <stdio.h>
2 int main()
3 {
4     int bil1 = 19, bil2 = 10, hasil;
5     hasil = bil1 + bil2;
6     printf("hasil penjumlahan %d dan %d adalah %d\n", bil1, bil2, hasil);
7     printf("hasil perkalian %d dan %d adalah %d\n", bil1, bil2, bil1*bil2);
8     return 0;
9 }
```

## Result

```
$gcc -o main *.c
```

```
$main
```

```
hasil penjumlahan 19 dan 10 adalah 29
```

```
hasil perkalian 19 dan 10 adalah 190
```

- Operator aritmatika digunakan pada baris ke-5 dan ke-7 pada program di atas

# Operator Modulus

```

1  #include <stdio.h>
2  int main()
3  {
4      int bil1, bil2, sisaBagi;
5      printf("Masukkan nilai bilangan 1: ");
6      scanf("%d", &bil1);
7      printf("Masukkan nilai bilangan 2: ");
8      scanf("%d", &bil2);
9      sisaBagi = bil1 % bil2;
10     printf("sisa pembagian %d dengan %d adalah %d\n", bil1, bil2, sisaBagi);
11     return 0;
12 }

```

```

Masukkan nilai bilangan 1: 15
Masukkan nilai bilangan 2: 3
sisa pembagian 15 dengan 3 adalah 0

```

Contoh 1

```

Masukkan nilai bilangan 1: 15
Masukkan nilai bilangan 2: 4
sisa pembagian 15 dengan 4 adalah 3

```

Contoh 2

```

Masukkan nilai bilangan 1: 15
Masukkan nilai bilangan 2: 2
sisa pembagian 15 dengan 2 adalah 1

```

Contoh 3

- Operator aritmatika (modulus) digunakan pada baris ke-9 pada program di atas
- Program di atas diuji coba sebanyak 3 kali dengan nilai masukan yang berbeda-beda:
  - Contoh 1:  $15\%3 = 0$  □  $15 - (3*5) = 0$
  - Contoh 2:  $15\%4 = 3$  □  $15 - (4*3) = 3$
  - Contoh 3:  $15\%2 = 1$  □  $15 - (2*7) = 1$



## Operator Penurunan (*decrement*) dan Peningkatan (*increment*)

- C menyediakan operator yang disebut sebagai operator peningkatan dan operator penurunan, yaitu :
  - ++ : operator untuk menaikkan nilai variabel sebesar 1 tingkat
  - -- : operator untuk menurunkan nilai variabel sebesar 1 tingkat
- Penempatan operator terhadap variabel dapat dilakukan di awal atau di belakang nama variabel. Hal ini bergantung pada kondisi yang dibutuhkan. Contoh:
  - $x = x+1$ ; □ dapat dituliskan sebagai ++x (*pre increment*) atau x++ (*post increment*)
  - $y = y-1$ ; □ dapat dituliskan sebagai --y (*pre decrement*) atau y-- (*post decrement*)





# Operator Penurunan (*decrement*) dan Peningkatan (*increment*)

```
1 #include <stdio.h>
2 int main()
3 {
4     int count = 0, loop;
5     loop = ++count;
6     //count=count+1; loop=count;
7     printf("loop = %d, count = %d\n", loop, count);
8     loop = count++;
9     //loop=count; count=count+1;
10    printf("loop = %d, count = %d\n", loop, count);
11 }
```

## Result

```
$gcc -o main *.c
$main
loop = 1, count = 1
loop = 1, count = 2
```

- Operator peningkatan digunakan pada baris ke-5 dan ke-8 pada program di atas.
- Baris ke-5 menggunakan *pre increment*, sehingga nilai variabel `count` dinaikkan terlebih dahulu sebelum dimasukkan ke dalam variabel `loop`.
- Sedangkan, baris ke-8 menggunakan *post increment*, sehingga nilai variabel `count` dimasukkan ke dalam variabel `loop` terlebih dahulu. Setelah itu, nilai variabel `count` dinaikkan 1 tingkat.





# Operator Penugasan (*Assignment*)

- Operator penugasan digunakan untuk memindahkan nilai dari suatu ungkapan (*expression*) ke suatu variabel.
- Operator penugasan yang umum digunakan dalam bahasa pemrograman, termasuk bahasa C adalah operator sama dengan (=).
- Contoh : `fahrenheit = celcius * 1.8 + 32;`
- Maka '=' adalah operator penugasan yang akan memberikan nilai dari hasil perhitungan : `celcius * 1.8 + 32` kepada variabel `fahrenheit`.
- Ekspresi di sebelah kanan tanda '=' akan diproses sampai tuntas, kemudian hasilnya di-assign ke variabel di sebelah kirinya
- Bahasa C juga memungkinkan dibentuknya penugasan menggunakan operator pengerjaan jamak dengan bentuk sebagai berikut :  
`var1 = var2 = ... = ekspresi ;`
- Misalnya : `a = b = 15;`
- maka nilai variabel `a` akan sama dengan nilai variabel `b` yakni 15



# Operator Kombinasi (Pemendekan)

```
x += 2; kependekan dari x = x + 2;  
x -= 2; kependekan dari x = x - 2;  
x *= 2; kependekan dari x = x * 2;  
x /= 2; kependekan dari x = x / 2;  
x %= 2; kependekan dari x = x % 2;  
x <<= 2; kependekan dari x = x << 2;  
x >>= 2; kependekan dari x = x >> 2;  
x &= 2; kependekan dari x = x & 2;  
x |= 2; kependekan dari x = x | 2;  
x ^= 2; kependekan dari x = x ^ 2;
```

# Operasi Input Output (Operasi I/O)

- Bahasa C sudah menyediakan beberapa fungsi standar yang bisa digunakan untuk operasi output dan input. Untuk bisa menggunakan fungsi standar operasi I/O, user perlu menuliskan header file `stdio.h` pada posisi teratas (bagian *preprocessor*) program.

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

- Fungsi Operasi Input:

- `scanf()` □ menerima input dari keyboard dengan format tertentu
- `getchar()` □ menerima input 1 karakter dari keyboard
- `gets()` □ menerima input string dari keyboard

- Fungsi Operasi Output:

- `printf()` □ menampilkan output ke layar dengan format tertentu
- `putchar()` □ menampilkan output berupa 1 karakter ke layar
- `puts()` □ menampilkan output berupa string yang diakhiri dengan enter ke layar



# Operasi Output – `printf()`

- Bentuk umum:  
`printf("format_tampilan", argumen);`
- `argumen` adalah data yang akan ditampilkan ke layar. Data dapat berupa variabel, konstanta atau bahkan sebuah ekspresi.
- `format_tampilan` yang digunakan tergantung dari tipe data yang digunakan oleh argumen yang ingin ditampilkan ke layar.
- Contoh:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int suhu = 10;
6     char skala = 'R';
7     printf("%d %c", suhu, skala);
8
9     return 0;
10 }
```

## Result

```
$gcc -o main *.c
$main
10 R
```



# Operasi Output – `printf()`

- Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      const float PI = 3.14f;
6      float r = 7;
7      printf("Luas lingkaran dg r = %f dan PI = %f adalah %f", r, PI, r*PI*PI);
8
9      return 0;
10 }
```

## Result

```
$gcc -o main *.c
```

```
$main
```

```
Luas lingkaran dg r = 7.000000 dan PI = 3.140000 adalah 69.017204
```

# Operasi Output – `printf()`

- Teks apapun yang ada di antara 2 tanda petik ganda “ “, akan dicetak TANPA MODIFIKASI, kecuali apabila teks tersebut mengandung karakter `%` atau `\`
- Karakter `%` menandakan format tampilan sedangkan karakter `\` menandakan karakter khusus yang harus diterjemahkan terlebih dahulu sebelum ditampilkan (*escape character*)
- Gabungan dari *escape character* dan karakter tertentu disebut sebagai *escape sequence*. Tiap *escape sequence* memiliki makna yang berbeda-beda.

# *Escape Sequence*

- `\n` : menyatakan karakter baris-baru
- `\"` : menyatakan karakter petik-ganda
- `'` : menyatakan karakter petik-tunggal
- `\\` : menyatakan karakter backslash
- `\t` : menyatakan karakter tab
- dsb



# Escape Sequence

- Contoh:

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int nrp = 2110191089;
6      char nama[] = "Anastasia";
7      int umur = 21;
8      printf("NRP = %d ", nrp);
9      printf("Nama = %s ", nama);
10     printf("Umur = %d ", umur);
11
12     return 0;
13 }

```

## Result

```
$gcc -o main *.c
```

Tanpa *Escape Sequence* New Line (\n)

```
$main
```

```
NRP = 2110191089 Nama = Anastasia Umur = 21
```

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int nrp = 2110191089;
6      char nama[] = "Anastasia";
7      int umur = 21;
8      printf("NRP = %d\n", nrp);
9      printf("Nama = %s\n", nama);
10     printf("Umur = %d\n", umur);
11
12     return 0;
13 }

```

## Result

```
$gcc -o main *.c
```

```
$main
```

```
NRP = 2110191089
```

```
Nama = Anastasia
```

```
Umur = 21
```

Dengan *Escape Sequence* New Line (\n)



# Format Tampilan

|              |                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %u           | untuk menampilkan data bilangan tak bertanda ( <i>unsigned</i> ) dalam bentuk desimal.                                                                                               |
| %d }<br>%i } | untuk menampilkan bilangan integer bertanda ( <i>signed</i> ) dalam bentuk desimal                                                                                                   |
| %o           | untuk menampilkan bilangan bulat tak bertanda dalam bentuk oktal.                                                                                                                    |
| %x }<br>%X } | untuk menampilkan bilangan bulat tak bertanda dalam bentuk heksadesimal<br>(%x → notasi yang dipakai : a, b, c, d, e dan f sedangkan %X → notasi yang dipakai : A, B, C, D, E dan F) |
| %f           | untuk menampilkan bilangan real dalam notasi : dddd.dddddd                                                                                                                           |
| %e }<br>%E } | untuk menampilkan bilangan real dalam notasi eksponensial                                                                                                                            |
| %g }<br>%G } | untuk menampilkan bilangan real dalam bentuk notasi seperti %f, %E atau %F<br>bergantung pada kepresisian data (digit 0 yang tak berarti tak akan ditampilkan)                       |
| l            | merupakan awalan yang digunakan untuk %d, %u, %x, %X, %o untuk menyatakan long int (misal %ld). Jika diterapkan bersama %e, %E, %f, %F, %g atau %G akan menyatakan <i>double</i>     |
| L            | Merupakan awalan yang digunakan untuk %f, %e, %E, %g dan %G untuk menyatakan <i>long double</i>                                                                                      |
| h            | Merupakan awalan yang digunakan untuk %d, %i, %o, %u, %x, atau %X, untuk menyatakan <i>short int</i> .                                                                               |

# Format Tampilan

- Penentuan panjang medan bagi tampilan data, dengan cara menyisipkan bilangan bulat sesudah tanda % dalam penentu format yang menyatakan panjang medan.

- Untuk data yang berupa bilangan bulat, misalnya:

```
int abad = 20;
printf("Abad %4d", abad);
```

- %4d menyatakan medan untuk menampilkan nilai dari variabel `abad` adalah sepanjang 4 karakter.
- Untuk data yang berupa bilangan real, spesifikasi medannya berupa `m.n`, dimana `m` = panjang medan dan `n` = jumlah digit pecahan

```
1 #include <stdio.h>
2
3 int main()
4 {
5     float desimal = 34.5671829f;
6     printf("bilangan desimal adalah %5.2f", desimal);
7
8     return 0;
9 }
```

## Result

```
$gcc -o main *.c
$main
bilangan desimal adalah 34.57
```



# Format Tampilan

- Untuk data yang berupa string, contoh : `printf("%12s", "Bahasa C");`

- Hasilnya : 

|  |  |  |  |   |   |   |   |   |   |  |   |
|--|--|--|--|---|---|---|---|---|---|--|---|
|  |  |  |  | B | a | h | a | s | a |  | C |
|--|--|--|--|---|---|---|---|---|---|--|---|

- Penentu format yang mengandung panjang medan, default-nya menampilkan data berbentuk rata kanan terhadap panjang medan yang diberikan.

- Untuk menampilkan dalam bentuk rata kiri, maka sesudah tanda % pada penentu format perlu disisipkan tanda – (minus),

- contoh : `printf("%-12s", "Bahasa C");`

- Hasilnya : 

|   |   |   |   |   |   |  |   |  |  |  |  |
|---|---|---|---|---|---|--|---|--|--|--|--|
| B | a | h | a | s | a |  | C |  |  |  |  |
|---|---|---|---|---|---|--|---|--|--|--|--|

# Operasi Output – puts ()

- Bentuk umum:  
`puts("teks_yang_akan_ditampilkan");`
- Fungsi `puts ()` hanya bisa digunakan untuk menampilkan teks / String. Secara otomatis fungsi tersebut akan menambahkan escape sequence new line di akhir string.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     puts("Kalimat 1");
6     puts("Kalimat 2");
7     return 0;
8 }
```

## Result

```
$gcc -o main *.c
$main
Kalimat 1
Kalimat 2
```

# Operasi Output – putchar ( )

- Digunakan khusus untuk menampilkan sebuah karakter di layar.
- Penampilan karakter tidak diakhiri dengan perpindahan baris.
- Contoh :

```
putchar('A');
```

- Perintah di atas akan menghasilkan keluaran yang sama dengan

```
printf("%c", 'A');
```

```
1 #include <stdio.h>
2
3 int main()
4 {
5     putchar('A');
6     return 0;
7 }
```

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%c", 'A');
6     return 0;
7 }
```

## Result

```
$gcc -o main *.c
$main
A
```

# Operasi Input – scanf ( )

- Bentuk umum:

```
scanf("format_tampilan", daftar_argumen);
```

- Tiap fungsi `scanf ( )` dapat menerima lebih dari 1 argumen yang berupa alamat dari variabel.
- Untuk menyatakan alamat dari variabel, di depan variabel dapat ditambahkan tanda `&` (operator alamat).
- Contoh:

```
scanf("%f", &radius);
```

- Komputer akan membaca bilangan yang dimasukkan oleh user sebagai bilangan real dengan tipe `float (%f)` dan menempatkan bilangan tersebut ke alamat dari variabel `radius (&radius)`

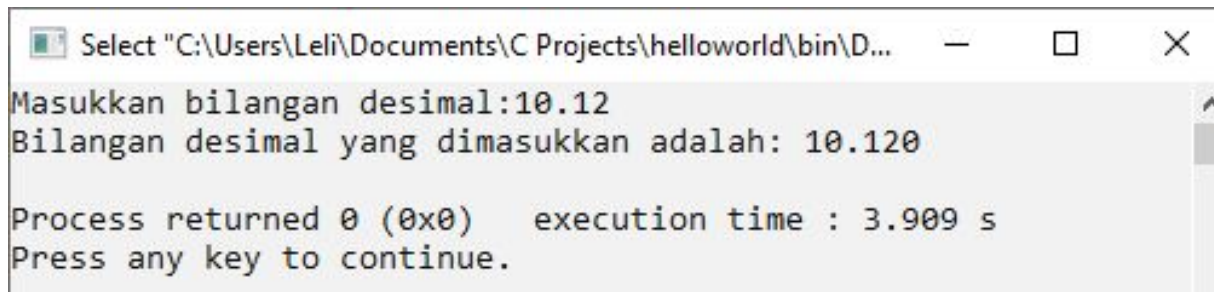




# Operasi Input – scanf ( )

- Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      float desimal;
6      printf("Masukkan bilangan desimal:");
7      scanf("%f", &desimal);
8      printf("Bilangan desimal yang dimasukkan adalah: %.3f\n", desimal);
9      return 0;
10 }
```



```
Select "C:\Users\Leli\Documents\C Projects\helloworld\bin\D...
Masukkan bilangan desimal:10.12
Bilangan desimal yang dimasukkan adalah: 10.120

Process returned 0 (0x0)   execution time : 3.909 s
Press any key to continue.
```

# Operasi Input – scanf ( )

- Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int dd, mm, yyyy;
6      printf("Masukkan tanggal dengan format: dd mm yyyy : ");
7      scanf("%d %d %d", &dd, &mm, &yyyy);
8      printf("Tanggal yang dimasukkan adalah: %d-%d-%d\n", dd, mm, yyyy);
9      return 0;
10 }
```

```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\...
Masukkan tanggal dengan format: dd mm yyyy : 12 1 2000
Tanggal yang dimasukkan adalah: 12-1-2000

Process returned 0 (0x0)   execution time : 6.066 s
Press any key to continue.
```



# Operasi Input – `getchar()`

- Bentuk umum:

```
nama_variabel = getchar();
```

- Digunakan khusus untuk menerima masukan berupa sebuah karakter dari keyboard

- Contoh:

```
kar = getchar();
```

- Perintah di atas akan memberikan hasil yang sama dengan perintah berikut:

```
scanf("%c", &kar);
```

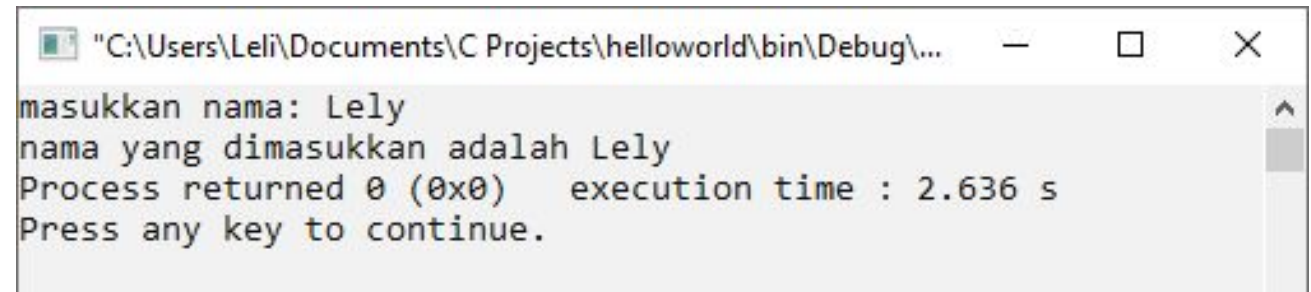
- maka variabel `kar` akan berisi karakter yang diketikkan oleh user atau EOF (end of file) jika ditemui akhir dari file



# Operasi Input – gets ()

- Bentuk umum:  
`gets(nama_variabel);`
- Digunakan khusus untuk menerima masukan beberapa karakter (String) dari keyboard. Input dari user akan disimpan di dalam array of char. Input berakhir apabila user menekan tombol enter.
- Contoh:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char nama [10];
6     printf("masukkan nama: ");
7     gets(nama);
8     printf("nama yang dimasukkan adalah %s", nama);
9 }
```



```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\...
masukkan nama: Lely
nama yang dimasukkan adalah Lely
Process returned 0 (0x0)   execution time : 2.636 s
Press any key to continue.
```

# bridge to the future

<http://www.eepis-its.edu>

