# STRING KARAKTER

contoh :

```
#include <stdio.h>
define DENSITY 62.4 /*human density (lbs/ft3) */
        float weight,volume;
        int     size,letters;
        char  name[40];
        printf (hi ! what's your first name ?.\n);
        scanf ("%s",name);
        printf ("%s, what's your weight in pounds ?\n",name);
        scanf ("%f", &weight);
        size       = sizeof name;
        letters   = strln(name);
        volume = weight/density;
        printf ("well,%s, your volume is %2.2f cubic feet .\n",name,volume);
        printf ("also, your first name has %d letters,\n",letters);
        printf ("and we have %d bytes to store iy in .\n",size);
```

OUTPUT :

```
        hi ! what's your name ?
        agus
        agus, what's your weight in pounds ?
        102.5
        well, agus, your volume is 1.64 cubic feet.
        also , your firts name has 4 letters;
        and we have 40 bytes to store it in.
```

HAL BARU :

Penggunaan array untuk string karakter yaitu : name
Penggunaan %s untuk menangani input dan output.
Penggunaan preprosesor C untuk mendefinisikan konstan simbolik density
Penggunaan fungsi C strlen() untuk mencari panjang string

**STRING KARAKTER**

sebuah string karakter adalah deretan satu atau lebih karakter, yang diakhiri oleh karakter akhir string..

Contoh : "saya sedang belajar string di C"
tanda petik ganda bukan termasuk bagian dari string, tapi hanya menandai string. Di C tidak ada tipe varaibel khusus untuk string.

string disimpan dalam array dari tipe character

| s | a | y | a | | s | e | d | a | n | g | | b | e | l | a | j | a | r | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| s | t | r | i | n | g | | d | i | | C | | | | | | | | | |

C menggunakan \0 atau karakter null pada posisi terakhir dari array untuk menandai akhir string, oki array mempunyai minimum 1 sel labih dari jumlah karakter yang disimpan. Contoh : 33 karakter pada string tersebut
Array adalah rangkaian terurut dari elemen data dari tipe yang sama.
Contoh : kita membuat array dari 40 sel memori , masing - masing menyimpan satu harga bertipe character.
Pendeklarasiannya adalah sbb : char name[40];
[ ] menunjukkan bahwa name adalah sebuah array, 40 menunjukkan jumlah elemen dan karakter mengidentifikasikan tipe dari masing - masing elemen.


char name[3] me-link 3 obyek data bertipe character.

meskipun kelihatannya kompleks, membuat array mengisikan karakter ke string satu per satu dan menambahkan sebuah \0 pada akhir dari string, tapi komputer pada prakteknya menangani detail ini.

```
/* praise*/
#define PRAISE "my sakes, that's a grand name !"
main()
{
        char name[50];
        printf ("what's your name ?\n);
        scanf ("%s", name);
        printf ("hello, %s. %s\n",name,PRAISE);
}
```

OUTPUT :

```
        what's your name ?
        elmo
        hello,elmo. my sakes, that's a grand name !
```

Perhatikan bahwa baik pada waktu memberi masukan variabel name menggunakaan scanf() maupun pada waktu mendefinisikan PRAISE kita tidak menambahkan \0 pada akhir string tapi komputer yang menambahkannya.

.

scanf() hanya membaca first name dari elmo; scanf() berhenti membaca input pada 'white space' pertama yang ditemukan (blank, tab, newline)
Jadi scanf() pada dasarnya hanya membaca satu kata .
C mempunyai fungsi lain seperti gets() untuk menangani masalah string yang lebih general.

"x" tidak sama dengan 'x' , karena :

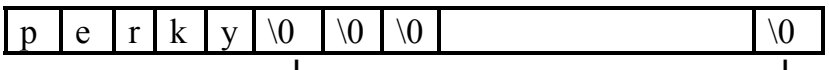
1. 'x' adalah tipe data dasar yi karakter sedang
   "x" adalah tipe turunan yi array dari karakter.
2. "x" sebenarnya mempunyai 2 karakter yi 'x' dan '\0.'

**PANJANG STRING - strlen()**
Fungsi strlen() memberikan panjang dari string dalam karakter

```
/* praise2 */
#define PRAISE "my sakes, that's a grand name !"
main()
{
        char name[50];
        printf ("what's your name ?\n);
        scanf ("%s", name);
        printf ("hello, %s. %s\n",name,PRAISE);
        printf ("your name of %d letters occupies %d memory cells.\n",
        strlen(name),sizeof name);
        printf ("the phrase of praise has %d letters ",strlen(PRAISE));
        printf ("and occupies %d memory cells.\n",sizeof PRAISE);
}
```

OUTPUT :

```
        what's your name ?
        perky
        hello,perky. My sakes, that's a grand name !
        your name of 5 letters occupies 50 memory cells.
        the phrase of praise has 30 lettersand occupies.31 memory cells
```




strlen() berhenti disini sizeof berhenti disini

Pada variabel name yang mempunyai 50 sel memori, maka sizeof melaporkan jumlah sel tersebut, sedangkan strlen() berhenti menghitung bila bertemu \0.

Pada PRAISE , strlen() memberi jumlah karakter yang sesungguhnya pada string termasuk spasi, sedang sizeof memberikan angka yang satu lebih besar karena \0 ikut dihitung.

Penggunaan sizeof yang berbeda

| Pada tipe data | pada variabel khusus |
|---|---|
| sizeof(char) | sizeof name |
| sizeof(float) | sizeof 6.28 |

**KONSTANTA dan PREPROSESOR C**

Mengapa kita memerlukan konstanta simbolik ???. Dengan adanya nama program lebih mudah dibaca.

```
Contoh :    corc = 3.14 * diameter;
            circ  = pi * diameter;      /*lebih dapat dibaca*/
```

Bila kita menggunakan satu konstanta pada beberapa tempat , kemudian kita ingin mengubahnya maka kita harus mencari dan mengubah satu per satu.
Bagaimana kita mengeset suatu konstanta simbolik ? . Deklarasikan sebagai suatu variabel dan beri nilai konstanta yang dikehendaki .
Contoh :

```
float pi;
pi = 3.14;
```

Untuk program kecil OK

**PREPROSESOR C**

#define PI 3.14
=> compile time substitusion
=> gunakan huruf besar untuk nama konstanta simbolis
=> bisa digunakan untuk konstanta char dan string

```
#define BEEP \'007'
#define ESS 'S'
#define OOPS "now you have done it !"
```

=> kumpulan dari statement #define pada satu file, misal const h bisa dipakai untuk program lain menggunakan #include "const.h" pada header
=> bagian dari program dalam tanda petik ganda adalah umum terhadap substitusi.

```
#define S 'student"
printf ("i am a S.\n");
```

OUTPUT :

```
i am a student.
```

**PRINTF()**

Fungsi yang paling umum digunakan dalam menampilkan data.Berbagai jenis data dapat ditampilkan ke layar dengan memakai fungsi ini.

Format :

printf( "string control",argumen1, argumen2,...)
string control adalah keterangan dalam tanda " " yang akan ditampilkan pada layar beserta identifier (spt %d, %f)
string control terdiri dari 2 bentuk informasi
1. Karakter-karakter yang akan di cetak secara literal.
2. Data identifier = conversion specification.

argumen1, argumen2 dll adl sesuatu yg akan mensubstitusi identifier bisa berupa.
1. variabel, atau
2. konstanta, atau
3. ekspresi / ungkapan yang dievaluasi dahulu sebelum nilainya dicetak.

Contoh :

```
printf ("%d",70);      argumen berupa konstanta
printf ("%d",a);       argumen berupa variabel
printf ("%d",a+70);    argumen berupa ungkapan

main()
{
        printf ("saya belajar C");                          string
        printf ('a');                                       character
        printf ("dua ditambah dua sama dengan %d",4); variabel
```

```
#include <stdio.h>
main()
{
        unsigned int segmen = 0xb880;
        printf ("nilai segmen grafik (oktal)                    : %o\n",segmen);
        printf ("nilai segmen grafik (desimal)          : %u\n",segmen);
        printf ("nilai segmen grafik (heksadesimal)     : %x\n",segmen);
}
```

OUTPUT :

```
nilai segmen grafik (oktal)             :134200
nilai segmen grafik (desimal)           : 47232
nilai segmen grafik (heksadesimal)      : b880
```

Perbedaan penentu format %g, %e dan %f ditunjukkan pada contoh berikut:

```
#include <stdio.h>

main()
{
        float x = 251000.0;
        printf ("format e = %e",x);
        printf ("format f =  %f",x);
        printf ("format g = %g",x);
}
```

OUTPUT :

```
        format e = 2.51000e+05
        format f  = 251000.000000
        format g  = 251000
```

Untuk nilai real, spesifikasi medan berupa : m,n dengan m menyatakan panjang medan dan n menyatakan jumlah digit pecahan.
Contoh :

```
        printf ("harga : Rp %8.2f \n",500.0);
```

| h | a | r | g | a |  | : |  | R | p |  |  |  | 5 | 0 | 0 | . | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

%8.2f menyatakan bahwa panjang medan dari bilangan real yang kan ditampilkan adalah 8 karakter dengan jumlah digit pecahan 2 buah.

| IDENTIFIER | OUTPUT |
|---|---|
| %d | integer bertanda dalam bentuk desimal |
| %c | karakter tunggal |
| %s | string |
| %e | bil.floating, notasi dengan e (eksponensial) |
| %f | bil.floating,notasi desimal |
| %g | bilangan floating / real, gunakan %f atau %e |
| %u | integer desimal, unsigned |
| %o | integer oktal unsigned |
| %x | integer heksadesimal unsigned |

Untuk data bertipe long, cukup menambah awalan l didepan tipe aslinya.

```
Misal : long integer              %ld
        unsigned octal integer    %l0
        unsigned hexa integer     %lx
```

Satu hal yang penting diperhatikan adalah urutan dari letak identifier/penentu format harus sesuai dengan urutan data yang akan mengisi identifier tsb.

```
printf ("%c merupakan abjad ke %d",'b',2);
```

identifier untuk 'b' adalah %c sedang untuk 2 adalah %d karena %c mendahului %d maka 'b' harus diletakkan di depan 2.

**PENENTU LEBAR FIELD (FIELD WIDTH SPECIFIER)**

```
main()
{
        float bil=2.5 , nomor = 30.756;
        clrscr();
        printf ('bilangan = %f \n",bil);
        printf ("nomor    = %f",nomor);
}
```

OUTPUT:

```
bilangan = 2.500000
nomor    = 30.756001
```

Desimal dan panjang field yang diberikan sebenarnya dapat diatur. cara mengatur lebar field dan jumlah desimal yang ingin dicetak cukup dengan memberikan format tambahan pada %f .

Contoh :

```
main()
{
        float bil = 2.5, nomor = 30.756;
        clrscr();
        printf ("bilangan = %10.2f \n",bil);
        printf ("nomor    = %10.2f, nomor);
}
```

OUTPUT :

```
bilangan  = ------  2.50
nomor     = ----- 30.76
```

Bila jumlah desimal yang ada lebih panjang dari yang akan dicetak, maka desimal tsb akan dibulatkan ke angka terdekat dapat dibulatkan ke atas atau ke bawah.
Pengaturan lebar field dapat juga diabaikan dengan pengertian hanya jumlah angka di belakang komanya saja yang diperhatikan.

Contoh :

```
main()
{
        float bil = 2.5, nomor = 30.756;
        clrscr();
        printf ("bilangan  = %.2f \n",bil);
        printf ("nomor     = %.2f, nomor);
}
```

OUTPUT :

```
bilangan  = 2.50----
nomor     = 30.76---
```

Data dapat juga dicetak dalam format rata kiri, dengan menyisipkan tanda - (minus) pada format tambahan tadi.

Contoh :

```
main()
{
        float bil = 2.5, nomor = 30.756;
        clrscr();
        printf ("bilangan = %-10.2f \n",bil);
        printf ("nomor    = %-10.2f, nomor);
}
```

OUTPUT :

```
bilangan  =  2.50----
nomor     =  30.76---
```

**CONVERSION SPECIFICATION MODIFIER**

| MODIFIER | ARTI |
|---|---|
| - (negatif) | item akan dicetak bermula pada sebelah kiri dari lebar field. Tanpa modifier ini item akan dicetak ke kanan. %-10d |
| string digit | Lebar field minimum. Field yang lebih lebar akan digunakan jika string atau angka yang dicetak tidak termuat %4d. |
| .string digit | Yaitu menyatakan presisi. Untuk tipe floating, jumlah digit yang akan dicetak disebelah kanan titik desimal.%.2f |
| l | berkoresponden dengan long, bukan int . %ld |

Contoh :

```
main()
{
        printf("/%d/\n", 33336);
        printf("/%2d/\336);
        printf("/%10d/\n", 336);
        printf("/%-10d/\n", 336
        printf("/%f/\n", 1234.56);
        printf("/%e/\n", 1234.56);
        printf("/%4.2f/\n", 1234.56);
        printf("/%3.1f/\n", 1234.56);
        printf("/%10.3f/\n", 1234.56);
        printf("/%10.3e/\n", 1234.56);
}
```

OUTPUT :

```
/336/
/336/
/       336/
/336        /
/1234.560059/
/1.234560E+03/
/1234.56/
/1234.6/
/      1234.560/
/      1..234E+03/
```

```
main()
{
        printf ("/% 10.35/","kemudian");
        printf ("/%-10.35/","kemudian");
}
```

OUTPUT:

```
        /-------kem/
        /kem-------/
```

```
# define BLURB "outstanding acting !"
main()
{
        print("/%25/\n", BLURB);
        printf ("/%225/\n",BLURB);
        print("/%22.5s/\n", BLURB);
        printf ("/%-22.5s/\n",BLURB);
}
```

OUTPUT :

```
        /outstanding acting !/
        /_ outstanding acting !/
        /                    outst/
        /outst                    /
```

```
main()
{
        printf ("%d",336);
        printf ("%c",65);
        printf ("%o",16);
        printf ("%x",32);
}
```

OUTPUT :

```
        336
        A       (ASCII)
        20      (oktal)
        20      (hexa)
```

**SCANF()**

Fungsi scanf() merupakan fungsi yang dapat digunakan untuk membaca data dari keyboard dan memasukkan ke dalam program. Fungsi scanf() hampir sama dengan fungsi printf(). Identifier yang digunakan dalam scanf() umumnya sama dengan yang digunakan dalam printf(), sedikit perbedaan yaitu scanf() tidak mengenal penggunaan %g.Fungsi scanf() dan printf() memungkinkan terjadinya komunikasi 2 arah dengan komputer, programnya disebut Interaktif. Daftar argumen dapat berupa satu atau beberapa argumen dan haruslah berupa alamat untuk menyatakan suatu alamat dari variabel di depan dapat ditambahkan tanda & ( tanda & dinamakan sebagai operator alamat ). & dikenal sebagai operator alamat (address operator). Contoh : scanf ("%f",&radius). Berarti "baca sebuah bilangan real (%f) dan tempatkan ke alamat dari radius (&radius).

Satu hal penting yaitu scanf() tidak dapat menggunakan pengaturan lebar field dan jumlah desimal.

Contoh :
```
printf("masukkan bil. pertama : ");
scanf() ("%10.2f",&bil);
```

Scanf() menggunakan pointer ke variabel
1. jika variabelnya bertipe data dasar, maka gunakan &
2. jika variabelnya adalah string, tidak menggunakan &

scanf() menggunakan white space (spasi, \n, \t) untuk memisahkan field.
untuk conversion spesification :
1. tidak ada %g
2. %f dan %e adalah ekuivalen
3. terdapat option %h untuk membaca short int.

**MEMASUKKAN BEBERAPA DATA SEKALIGUS DALAM SATU BARIS**

Dengan fungsi scanf() mengijinkan memasukkan beberapa data sekaligus dalam satu baris selama jumlah serta tipe data yang dimasukkan sesuai dengan identifier yang diberikan dalam scanf(). Data yan akan dimasukkan dapat dipisahkan dengan spasi, tab atau dengan tanda pemisah lain seperti koma, garis hubung, titik dua. Pemisah data dalam input yang diketikkan harus sama dengan pemisah data yang digunakan dalam scanf()

Contoh :
```
printf( "masukkan 3 bilangan bulat :");
scanf("%d  %d  %d ",&bil1,&bil2,&bil3);

OUTPUT :
masukkan 3 bilangan bulat : 12  30  8
```

```
main()
{
        int     age;
        float assets;
        char  pet[30];
        printf( "enter your age, assets, and favorite pet.\n");
        printf (%d %f", &age, &assets);
        printf ("%s", pet);
        printf ("%d $%.of  %s \n",age, assets, pet);
}
```

```
#include <stdio.h>
main()
{
        float dolar,rupiah;
        char  bel;
        bel = '\007';    /*character untuk bunyi bel */
        printf ("berapa dollar uang anda ?\n");
        scanf ("%f", &dolar);
        rupiah = dolar * 2000.00;
        printf ("%c uang anda setara dengan ",bel);
        printf ("%2.2f rupiah %c \n",rupiah,bel);
}
```

OUTPUT :

```
        berapa dollar uang anda ?
        1.2
        uang anda setara dengan 2400.00 rupiah.
```

```
#include,stdio.h>
#define PI 3.141593
main()
{
        float radius, keliling, luas;
        printf ("masukkan data jari - jari lingkaran : ");
        scanf (%f", &radius);
        keliling = 2 * PI * radius;
        luas      = PI * radius * radius;
        printf ("Data lingkaran :\n);
        printf ("jari - jari = %f \n",radius);
        printf ("keliling    = %f \n",keliling);
        printf ("luas       = %f \n",luas);
}
```

OUTPUT :

```
        masukkan  data jari - jari : 5
        data lingkaran          = 5
        keliling                = 31.415930
        luas                    = 78.539825
```