Bab 9
String

String merupakan sekumpulan karakter. Dalam bahasa C, string direpresentasikan dengan *array of char* yang diakhiri dengan karakter *null* (null terminated string).

Deklarasi String
String dedeklarasikan dengan membuat array of char :
`char <nama variable>[<panjang string>]`

Contoh deklarasi string :
```
char judul[50];
char nama_kota[20];
```

Karena string dibentuk dari *array of char*, maka panjang string bisa tak terbatas. Tentu saja panjang string dibatasi oleh jumlah memori pada komputer anda =), namun karena memori komputer sekarang amat besar, dapat kita katakan panjang string “tidak terbatas”.

Walaupun sebuah string pada bahasa C “tidak terbatas”, kita tetap perlu memberitahukan komputer berapa panjang string yang ingin kita buat. Ini diperlukan supaya komputer dapat “memesankan” tempat yang cukup luas pada memori untuk menyimpan string tersebut.

Misalnya pada deklarasi string `char judul[50]`, kita memesan tempat sebanyak 50 karakter untuk variable `judul`. Setelah memesan tempat, kita harus menaatinya! Kita tidak boleh sembarangan mengisi variable tersebut lebih banyak dari tempat yang tersedia, karena dapat mengganggu isi variable lain. Namun tentu saja kita boleh mengisi lebih sedikit dari tempat yang tersedia =).

Satu hal yang perlu diperhatikan, string pada bahasa C diakhiri dengan sebuah karakter *null*, karena itu kita perlu memesankan tempat untuk karakter *ini*. Misalnya kita ingin membuat sebuah string untuk menyimpan data NPM. Ukuran NPM kita adalah 10 karakter. Jika kita mendeklarasikan sebuah string :
```
char NPM[10];
```
maka tidak akan ada tempat untuk menyimpan karakter *null*.

Untuk menyimpan string sepanjang 10, kita perlu menyediakan tempat sepanjang 11 karakter :
```
char NPM[11];
```

Tentu saja kita boleh membuat string dengan panjang 20, 100, atau bahkan ribuan karakter untuk menyimpan NPM. Selama panjang string lebih dari 11 boleh-boleh saja, tapi tentunya hal itu menjadi pemborosan memori =(.

Tips
Jika memang kita sudah tahu berapa ukuran data yang akan disimpan, ada baiknya kita mendeklarasikan variable dengan ukuran secukupnya =).

Input / Output String
Input string pada bahasa C sedikit berbeda dari cara input variable-variable lainnya. Ini dikarenakan string pada bahasa C sebenarnya hanya sebuah array.

Untuk input dengan `scanf`, kita menggunakan *formatting* `%s`. Untuk input string, kita tidak perlu menggunakan tanda `&`. Perhatikan contoh berikut :

```
char nama[50];
printf("masukan nama :");
scanf("%s", nama);
```

Sama halnya dengan input, untuk output dengan `printf` kita gunakan *formatting* `%s` juga. Perhatikan contoh berikut :

```
char nama[50];
printf("masukan nama :");
scanf("%s", nama);
printf("Selamat siang %s !", nama);
```

Formatting string `%s` pada `scanf` memiliki sebuah keterbatasan. Cara ini hanya dapat membaca input sebuah kata. Misalnya saya memasukan nama "`Joanna Helga`", maka isi variable `nama` adalah "`Joanna`" saja.

Untuk membaca input beberapa kata, gunakan formatting `%[^\n]`. *Formatting string* ini berarti "bacalah hingga menemukan \n", yang dengan kata lain adalah "baca hingga akhir baris". Perhatikan contoh berikut ini :

```
char nama[50];
printf("masukan nama :");
scanf("%[^\n]", nama);
printf("Selamat siang %s !", nama);
```

Untuk membaca beberapa baris string, jangan lupa menambahkan karakter \n pada scanf untuk membuang tanda pindah baris (enter). Perhatikan contoh berikut :

```
char nama[50];
char alamat[100];
printf("masukan nama : ");
scanf("%[^\n]", nama);
printf("masukan alamat : ");
scanf("\n%[^\n]", alamat);
printf("Hai %s ! Alamat anda di %s",nama, alamat);
```

Kita dapat menggabungkan penggunaan `%s` dan `%[^\n]` ini sesuai kebutuhan. Misalnya kita ingin meng-input nama depan, nama belakang, dan alamat, maka kita dapat membuat program seperti berikut :

```
char nama_depan[50], nama_belakang[50];
char alamat[100];

printf("masukan nama depan & belakang: ");
scanf("%s %s", nama_depan, nama_belakang);
printf("masukan alamat: ");
scanf("\n%[^\n]", alamat);

printf("Hallo %s %s !\n", nama_depan, nama_belakang);
printf("Alamat anda di %s\n", alamat);
```

Program diatas akan memisahkan input nama kedalam 2 variable, yaitu `nama_depan` dan `nama_belakang`. Pemisahan ini dilakukan berdasarkan penempatan spasi, karena formatting string `%s` dipisahkan dengan spasi. Misalnya saya memasukan nama "`Joanna Helga`", maka "`Joanna`" akan masuk ke variable `nama_depan`, dan "`Helga`" akan masuk ke variable `nama_belakang`. Tapi misalnya saya memasukan nama "`JoannaH elga`", maka komputer akan menganggap "`JoannaH`" sebagai `nama_depan` dan "`elga`" sebagai `nama_belakang`.

Assignment dan Operasi pada String

Assignment atau pemberian nilai pada variable string juga sedikit berbeda dari variable-variable lain. Hal ini juga dikarenaka string pada bahasa C sebenarnya merupakan array.

Jika kita punya variable dengan tipe `int` misalnya, kita dapat menuliskan :

```
int harga;
harga = 3500;
```

Tapi untuk tipe string, kita tidak boleh menuliskan seperti ini :

```
char nama[50];
nama = "Joanna";
```

Sama halnya dengan operasi-operasi pada variable. Jika pada variable `int` kita dapat menuliskan seperti ini :

```
harga = harga – diskon;
```

Pada string kita tidak boleh menuliskan seperti ini :

```
nama = nama_depan + nama_belakang;
```

Jadi bagaimana caranya memanipulasi isi variable string ?

Cara yang cukup mudah adalah dengan menggunakan fungsi-fungsi standar bahasa C. Sebelum menggunakan fungsi-fungsi ini, kita perlu mendeklarasikannya terlebih dahulu dengan menuliskan **`#include<string.h>.`**

Fungsi-fungsi yang umum dipakai adalah sebagai berikut :

Sintaks	Kegunaan
`int length(char str1[]);`	• untuk mencari panjang string
`strcpy(char str1[], char str2[]);`	• untuk mengisikan string `str2` kedalam variable `str1`. • untuk menyalin isi variable `str2` ke variable `str1`.
`strcmp(char str1[], char str2[]);`	• untuk membandingkan nilai 2 buah string/variable. * Jika `str1` lebih kecil dari `str2`, maka nilai return-nya negatif. * Jika `str1` lebih besar dari `str2`, maka nilai return-nya positif. * Jika kedua string sama, maka nilai return-nya 0.
`strcat(char str1[], char str2[]);`	• untuk menyambung 2 buah string. Hasil string yang telah tersambung disimpan di `str1`.

Contoh penggunaan strcpy

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

int main()
{
  char str1[50];
  char str2[50];

  // mengisi str1 = "Hello World!"
  strcpy(str1, "Hello World!");
  // mengisi str2 = str1
  strcpy(str2, str1);

  printf("isi str1 : %s\n", str1);
  printf("isi str2 : %s\n", str2);

  return 0;
}
```

Contoh penggunaan strcmp

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

int main()
{
```

```
  char str1[50];
  char str2[50];

  // mengisi str1 = "Hello"
  strcpy(str1, "Hello");
  // mengisi str2 = "Hallo"
  strcpy(str2, "Hallo");

  printf("hasil strcmp : %d\n", strcmp(str1, str2));

  return 0;
}
```

Contoh penggunaan strcat

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

int main()
{
  char str1[50];
  char str2[50];

  // mengisi str1 = "Hello "
  strcpy(str1, "Hello ");
  // mengisi str2 = "World"
  strcpy(str2, "World");

  strcat(str1, str2);
  printf("isi str1 : %s\n", str1);
  printf("isi str2 : %s\n", str2);

  return 0;
}
```

Fungsi-fungsi lain yang cukup berguna adalah fungsi untuk mengubah karakter menjadi huruf kecil atau huruf kapital. Sayangnya fungsi ini hanya menerima input sebuah char, jadi jika kita butuh mengubah sebuah string kedalam huruf kecil / kapital, kita harus menggunakan bantuan perulangan.
Untuk menggunakan fungsi-fungsi ini, kita perlu mendefinisikan **`#include<ctype.h>`**

Sintaks	**Kegunaan**
`char tolower(char c)`	• mengembalikan huruf kecil dari c
`char touper(char c)`	• mengembalikan huruf kapital dari c

Contoh penggunaan tolower dan toupper

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

```
#include<ctype.h>

int main()
{
  char c;

  c = 'A';
  printf("Isi c mula-mula : %c\n", c);

  printf("Isi tolower(c) adalah : %c\n", tolower(c));
  printf("Isi toupper(c) adalah : %c\n", toupper(c));

  return 0;
}
```