BAB VII
STRING

Tujuan :

1. Menjelaskan tentang konsep string
2. Menjelaskan operasi I/O pada string.
3. Menjelaskan cara mengakses elemen string
4. Menjelaskan berbagai fungsi mengenai string

7.1 Konstanta dan Variabel String

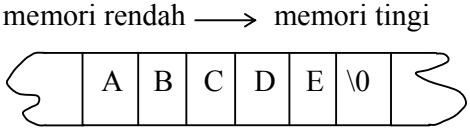
String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks, misalnya untuk menampung (menyimpan) suatu kalimat. Pada bahasa C, string bukanlah merupakan tipe data tersendiri, melainkan hanyalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk ***array*** berdimensi satu.

7.1.1 Konstanta String

Suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda, misalnya:

```
"ABCDE"
```

Nilai string ini disimpan dalam memori secara berurutan dengan komposisi sebagai berikut:



Gambar 7.1 Komposisi penyimpanan string dalam memori

Setiap karakter akan menempati memori sebesar 1 byte. Byte terakhir otomatis akan berisi karakter NULL (\0). Dengan mengetahui bahwa suatu string diakhiri nilai NULL, maka akhir dari nilai suatu string akan dapat dideteksi. Sebagai sebuah array karakter, karakter pertama dari nilai string mempunyai indeks ke-0, karakter kedua mempunyai indeks ke-1, dan seterusnya.

7.1.2 Variabel String

Variabel string adalah variabel yang dipakai utuk menyimpan nilai string. Misalnya :

```
char name[15];
```

merupakan instruksi untuk mendeklarasikan variabel string dengan panjang maksimal 15 karakter (termasuk karakter NULL). Deklarasi tersebut sebenarnya tidak lain merupakan deklarasi array bertipe *char* **.**

7.2 Inisialisasi String

Suatu variabel string dapat diinisialisasi seperti halnya array yang lain. Namun tentu saja elemen terakhirnya haruslah berupa karakter NULL. Sebagai contoh :

```
char name[] = {'R','I','N', 'I','\0'};
```

yang menyatakan bahwa **name** adalah variabel string dengan nilai awal berupa string : “RINI” . Bentuk inisialisasi yang lebih singkat :

```
char name[] = “RINI”;
```

Pada bentuk ini, karakter NULL tidak perlu ditulis. Secara implisit akan disisipkan oleh kompiler. Perlu diperhatikan, bila **name** dideklarasikan sebagai string, penugasan *(assignment)* suatu string ke variabel string seperti

```
name = “RINI”;
```

adalah **tidak diperkenankan.** Pengisian string ke variabel string akan dibahas pada sub bab berikutnya.

7.3 Input Output Data String

7.3.1 Memasukkan Data String

Pemasukan data string ke dalam suatu variabel biasa dilakukan dengan fungsi *gets()* atau *scanf()*. Bentuk umum pemakaiannya adalah sebagai berikut :

```
#include <stdio.h>
gets(nama_array);
```

atau

```
#include <stdio.h>
scanf("%s", nama_array);
```

Perhatikan :
- → **nama_array** adalah variabel bertipe *array of char* yang akan digunakan untuk menyimpan string masukan.
- → Di depan **nama_array** tidak perlu ada operator ***&*** (operator alamat), karena **nama_array** tanpa kurung siku sudah menyatakan alamat yang ditempati oleh elemen pertama dari *array* tsb.
- → Kalau memakai *scanf()*, data string masukan tidak boleh mengandung spasi.

Di bawah ini diberikan contoh program untuk memasukkan data nama seseorang ke dalam array bernama **name**.

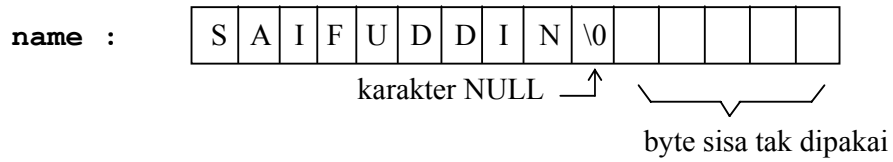
```
/* File program : yourname.c
Contoh memasukkan data string dari keyboard */

#include <stdio.h>

main()
{
     char name[15];

     printf("Masukkan nama Anda : ");
     //gets(name);
     scanf("%s", name);

     printf("\nHalo, %s.  Selamat belajar string.\n", name);
}
```

Contoh eksekusi :

```
Masukkan nama Anda :  SAIFUDDIN
Halo, SAIFUDDIN.  Selamat belajar string.
```

Ruang yang disediakan setelah deklarasi: `char name[15];`

Setelah data yang dimasukkan berupa : **SAIFUDDIN**



Gambar 7.2 Variabel string dan data string

Pada gambar di atas nama array tanpa kurung siku (**name**) menyatakan alamat dari lokasi data string. Dan dengan pemasukan data seperti di atas, lokasi memori yang terletak sesudah lokasi yang berisi '**N**' akan secara otomatis terisi karakter NULL (lihat gambar 7.2)

Perlu diketahui, fungsi *gets()* akan membaca seluruh karakter yang diketik melalui keyboard sampai tombol ENTER ditekan. Dalam hal ini tidak ada pengecekan terhadap batasan panjang array yang merupakan argumennya. Jika string yang dimasukkan melebihi ukuran array, maka sisa string (panjang string masukan dikurangi ukuran array plus karakter NULL) akan ditempatkan di lokasi sesudah bagian akhir dari array tersebut. Tentu saja kejadian seperti ini bisa menimbulkan hal yang tidak diinginkan, misalnya berubahnya isi variabel yang dideklarasikan sesudah array tersebut karena tertumpuki oleh string yang dimasukkan *(overwrite)*, atau perilaku program yang sama sekali berbeda dengan kemauan user yang dalam hal ini pelacakan kesalahannya *(debugging)* sangat sulit dilakukan, atau bahkan terjadi penghentian program secara tidak normal.

Untuk mengatasi hal itu, disarankan untuk menggunakan fungsi *fgets()* untuk menggantikan fungsi *gets()* dalam memasukkan data string.

Bentuk umum pemakaian *fgets()* adalah :

```
#include <stdio.h>
fgets(nama_array, sizeof nama_array, stdin);
```

Untuk lebih jelasnya, perhatikan contoh penggunaan fungsi *fgets()* ini pada contoh program untuk menghitung karakter masukan (**hitkar.c**) di halaman 118.

7.3.2 Menampilkan Isi Variabel String

Untuk menampilkan isi variabel string, fungsi yang digunakan adalah *puts()* atau *printf()*. Bentuk umum pemakaiannya adalah sebagai berikut :

```
#include <stdio.h>
puts(var_string);
```

atau

```
printf("%s",var_string);
```

Dalam hal ini **var_string** adalah sebuah variabel yang berupa sebuah *array of char*. Fungsi *puts()* akan menampilkan isi dari **var_string** dan secara otomatis menambahkan karakter '\n' di akhir string. Sedangkan fungsi *printf()* akan menampilkan isi variabel string tanpa memberikan tambahan '\n'. Sehingga, agar kedua pernyataan di atas memberikan keluaran yang sama, maka pada pernyataan *printf()* dirubah menjadi :

```
printf("%s\n", var_string);
```

Contoh program berikut akan menampilkan isi variabel **kompiler_c**, berdasarkan dua bentuk inisialisasi string yang dibahas pada sub bab 7.3

```
/* File program : initstr.c
Contoh inisialisasi string */

#include <stdio.h>

void bentuk1(void);
void bentuk2(void);

main()
{
     bentuk1();
     bentuk2();
}

void bentuk1(void)
{
     char kompiler_c[] =
     {'V','i','s','u','a','l',' ','C','+','+','\0'};

     puts(kompiler_c);
}
```

```
void bentuk2(void)
{
     char kompiler_c[] = "Visual C++";
     printf("%s\n", kompiler_c);
}
```

Contoh eksekusi :

```
Visual C++
Visual C++
```

7.4 Mengakses Elemen String

Variabel string merupakan bentuk khusus dari array bertipe *char*. Oleh karena itu, elemen dari variabel string dapat diakses seperti halnya pengaksesan elemen pada array. Program berikut menunjukkan cara mengakses elemen array untuk menghitung total karakter dari string yang dimasukkan melalui keyboard.

```
/* File Program : hitkar.c
Contoh untuk menghitung banyaknya karakter dari suatu string
yang dimasukkan melalui keyboard  */

#include <stdio.h>

#define MAKS 256

main()
{
     int i, jumkar = 0;
     char teks[MAKS];

     puts("Masukkan suatu kalimat (maks 255 karakter).");
     puts("Saya akan menghitung jumlah karakternya.\n");
     fgets(teks, sizeof teks, stdin); //masukan dr keyboard
     for(i=0; teks[i]; i++)
          jumkar++;

     printf("\nJumlah karakter = %d\n", jumkar);
}
```

Contoh eksekusi :

```
Masukkan satu kalimat (maks 255 karakter).
Saya akan menghitung jumlah karakternya.
```

```
SAYA SUKA BELAJAR BAHASA C

Jumlah karakter = 26
```

Perhitungan jumlah karakter dari string teks dapat dilakukan dengan memeriksa elemen dari string dimulai dari posisi yang pertama (indeks ke-0) sampai ditemukannya karakter NULL. Elemen yang ke-i dari teks dinyatakan dengan

```
teks[i]
```

Pemeriksaan terhadap **teks[i]** selama tidak berupa karakter NULL (dimulai dari indeks ke-0) dilakukan dengan instruksi

```
for(i=0; teks[i]; i++)
    jumkar++;
```

Kondisi **teks[i]** pada *for* mempunyai makna yang secara implisit berupa

```
teks[i] != '\0';
```

atau "karakter yang ke-i dari **teks** tidak sama dengan karakter NULL"

Contoh berikut ini akan memberikan gambaran mengenai cara menyalin nilai ke suatu variabel string.

```
/* File program : salinstr.c
   Contoh menyalin suatu string */

#include <stdio.h>

#define MAKS 30

main()
{
   int i;
   char asal[] = "Saya menyukai bahasa C";
   char hasil[MAKS];

   i=0;
   while (asal[i] != '\0')
     {
        hasil[i] = asal[i];
        i++;
     }
   hasil[i] = '\0'; /* beri karakter NULL */
   printf("Isi hasil : %s\n", hasil);
}
```

Contoh eksekusi :

```
Isi hasil : Saya menyukai bahasa C
```

Untuk menyalin isi variabel string **keterangan** ke **kalimat**, pernyataan yang digunakan berupa

```
   i=0;
   while (keterangan[i] != '\0')
     {
        kalimat[i] = keterangan[i];
        i++;
     }
```

Selama **keterangan[i]** tidaklah berupa karakter NULL, maka **keterangan[i]** disalin ke **kalimat[i]**. Jelas bahwa di dalam loop while tidak terdapat penyalinan karakter NULL dari **keterangan** ke **kalimat**. Oleh karena itu sekeluarnya dari loop while, pemberian karakter NULL ke **kalimat** perlu dilakukan. Pernyataan yang digunakan berupa

```
kalimat[i] = '\0';
```

Bentuk yang lebih singkat untuk melakukan penyalinan dari keterangan ke kalimat berupa

```
i=0;
while (kalimat[i] = keterangan[i])
     i++;
```

Dengan penulisan seperti di atas, penyalinan karakter NULL juga akan dilakukan secara otomatis. Setelah menyalin karakter NULL (karena kondisi bernilai NULL) maka eksekusi terhadap loop akan dihentikan. Dengan demikian sekeluarnya dari loop tidak perlu lagi ada pernyataan.

```
kalimat[i] = '\0';
```

7.5 Fungsi-Fungsi Mengenai String

Berikut ini akan dibahas beberapa fungsi pustaka yang umumnya disediakan oleh kompiler C untuk mengoperasikan suatu nilai string. Fungsi-fungsi pustaka untuk operasi string, prototype-prototype nya berada di file judul **string.h.** Beberapa di antara fungsi pustaka untuk operasi string akan dibahas di bawah ini.

7.5.1 Fungsi *strcpy()* untuk Menyalin Nilai String

Bentuk pemakaian :

```
#include <string.h>
strcpy(tujuan, asal)
```

Fungsi ini dipakai untuk menyalin string **asal** ke variabel string **tujuan** termasuk karakter '\0'. Keluaran dari fungsi ini *(return value)* adalah string **tujuan**. Dalam hal ini, variabel **tujuan** haruslah mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string **asal**. Contoh implementasinya bisa dilihat pada program **salinstr2.c** di bawah ini.

```
/* File program :salinstr2.c
Contoh menyalin isi str2 ke str1 */

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

#define MAKS 80
```

```
main()
{
     char str1[MAKS];
     char str2[]="ABCDE";

     strcpy(str1, str2);  /* menyalin isi str2 ke str1 */
     printf("String pertama adalah : %s\n", str1);
     printf("String kedua adalah   : %s\n", str2);
}
```

Contoh eksekusi :

```
String pertama adalah : ABCDE
String kedua adalah   : ABCDE
```

7.5.2 Fungsi *strlen()* untuk Mengetahui Panjang Nilai String

Bentuk pemakaian :

```
#include <string.h>
strlen(var_string);
```

Fungsi ini digunakan untuk memperoleh banyaknya karakter di dalam string yang menjadi argumennya (**var_string**). Keluaran dari fungsi ini adalah panjang dari **var_string**. Karakter NULL tidak ikut dihitung. Contoh implementasinya bisa dilihat pada program **panjangstr.c** di bawah ini.

```
/* File program : panjangstr.c
Contoh memperoleh panjang suatu string */

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

main()
{
   char salam[] = "Halo";

   printf("Panjang string = %d karakter\n",  strlen(salam));
}
```

Contoh eksekusi :

```
Panjang string = 4 karakter
```

7.5.3 Fungsi *strcat()* untuk Menggabung Nilai String

Bentuk pemakaian :

```
#include <string.h>
strcat(tujuan, sumber);
```

Menggabungkan dua buah nilai string tidak dapat dilakukan dengan operator '+', karena operator ini bukan operator untuk operasi string. Penggabungan dua buah nilai string dapat dilakukan dengan fungsi pustaka *strcat()* dengan menambahkan string **sumber** ke bagian akhir dari string **tujuan**. Keluaran dari fungsi ini adalah string **tujuan**. Contoh implementasinya bisa dilihat pada program **gabungstr.c** di bawah ini.

```
/* File program :gabungstr.c
  Contoh menggabungkan isi string1 dengan string2 */
#include <stdio.h>
#include <string.h>

#define PJG 15

main()
{
   char str1[PJG], str2[PJG];

   strcpy(str1, "sala");  /* str1 diisi "sala" */
   strcpy(str2, "tiga");  /* str2 diisi "tiga" */

   strcat(str1, str2);    /* tambahkan str2 ke akhir str1 */

   printf("str1 → %s     str2 → %s\n", str1, str2);
}
```

Contoh eksekusi :

```
str1 → salatiga     str2 → tiga
```

Dalam hal ini **str1** ("sala") digabungkan dengan **str2** ("tiga") dengan hasilnya berada di **str1** ("salatiga").

7.5.4 Fungsi *strcmp()* untuk Membandingkan Dua Nilai String

Membandingkan dua nilai string juga tidak dapat digunakan dengan operator hubungan, karena operator tersebut tidak untuk operasi string. Membandingkan dua buah nilai string dapat dilakukan dengan fungsi pustaka *strcmp()*.

Contoh bentuk pemakaian fungsi :

```
#include <string.h>
strcmp(str1, str2);
```

Fungsi ini dipakai untuk membandingkan string **str1** dengan string **str2**. Keluaran dari fungsi ini bertipe *int* yang berupa nilai :

- `-1`, jika **str1** kurang dari **str2**
- `0`, jika **str1** sama dengan **str2**
- `1`, jika **str1** lebih dari **str2**

Pembandingan dilakukan untuk karakter pada posisi yang sama dari **str1** dan **str2**, dimulai dari karakter terkiri. Acuan pembandingan dari dua buah karakter didasarkan oleh nilai ASCII-nya. Misal, karakter 'A' lebih kecil daripada 'B' dan karakter 'B lebih kecil daripada 'C'. Contoh implementasinya bisa dilihat pada program **bandingstr.c** di bawah ini.

```
/* File program :bandingstr.c
   Contoh membandingkan isi dua buah string */

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

main()
{
   char str1[]="HALO";
   char str2[]="Halo";
   char str3[]="HALO";

   printf("Hasil pembandingan %s dengan %s --> %d\n",
          str1, str2, strcmp(str1, str2));

   printf("Hasil pembandingan %s dengan %s --> %d\n",
          str2, str1, strcmp(str2, str1));

   printf("Hasil pembandingan %s dengan %s --> %d\n",
          str1, str3, strcmp(str1, str3));
}
```

Contoh eksekusi :

```
Hasil pembandingan HALO dengan Halo --> -1
Hasil pembandingan Halo dengan HALO --> 1
Hasil pembandingan HALO dengan HALO --> 0
```

7.5.5 Fungsi *strchr()* untuk Mencari Nilai Karakter dalam String

Bentuk pemakaian :

```
#include <string.h>
strchr(var_string, kar);
```

Fungsi ini dapat digunakan untuk mencari suatu nilai karakter yang berada dalam suatu nilai string. Dalam hal ini adalah mencari karakter **kar** dalam string **var_string**. Keluaran dari fungsi ini adalah alamat posisi dari karakter pertama pada nilai string, yang sama dengan karakter yang dicari. Jika karakter yang dicari tidak ada dalam nilai string, maka fungsi ini akan memberikan hasil nilai pointer kosong *(null)*. Contoh implementasinya bisa dilihat pada program **carikar.c** di bawah ini.

```
/* File program : carikar.c
   Contoh mencari karakter dalam sebuah string */

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

main()
{
   char str[]="ABcde";         /* inisialisasi string */
   char *hasil1,*hasil2;
   /* var bertipe pointer to char, agar bisa ditampilkan isi
   dari alamat yang ditunjuk oleh hasil1 & hasil2 */

   hasil1 = strchr(str, 'B');
   hasil2 = strchr(str, 'X');

   printf("Dari string ABcde\n");
   printf("Mencari karakter B = %s\n", hasil1);
   printf("Mencari karakter X = %s\n", hasil2);
}
```

Contoh eksekusi :

```
Dari string ABcde
Mencari karakter B = Bcde
Mencari karakter X = (null)
```

Contoh di atas menunjukkan penggunaan fungsi *strchr()* untuk mencari nilai karakter 'B' dan karakter 'X' dalam string 'ABcde'. Karakter 'B' ada dalam nilai string yang dicari, sehingga fungsi *strchr()* memberikan hasil alamat dari karakter B tersebut yang kemudian alamat ini disimpan dalam variabel pointer **hasil1**. Jika variabel pointer **hasil1** ini ditampilkan dengan menggunakan kode format untuk nilai string (`%s`), maka mulai dari alamat tersebut sampai dengan akhir dari nilai string yang bersangkutan akan ditampilkan, sehingga didapatkan keluaran :

```
Mencari karakter B = Bcde
```

Sedangkan pencarian karakter 'X' memberikan hasil null karena karakter tersebut tidak ditemukan dalam string 'ABcde", sehingga didapatkan keluaran :

```
Mencari karakter X = (null)
```

Keterangan lebih lanjut tentang pointer ini, akan dibahas pada bab VIII.

Kesimpulan :

- String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks.
- Pada bahasa C, string bukanlah merupakan tipe data tersendiri, melainkan hanyalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk array berdimensi satu
- Suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda
- Pemasukan data string ke dalam suatu variabel biasa dilakukan dengan fungsi *gets()* atau *scanf()*.
- Untuk menampilkan isi variabel string, fungsi yang digunakan adalah *puts()* atau *printf()*.
- Untuk mengoperasikan suatu nilai string ada beberapa fungsi pustaka yang prototype-prototype-nya berada di file judul **string.h**, sehingga dalam suatu program yang di dalamnya terdapat manipulasi string, haruslan ditambah : *#include <string.h>*.

- Beberapa fungsi untuk manipulasi string adalah sbb :
  a. Fungsi *strcpy()* untuk menyalin nilai string
  b. Fungsi *strlen()* untuk mengetahui panjang nilai string
  c. Fungsi *strcat()* untuk menggabung nilai string
  d. Fungsi *strcmp()* untuk membandingkan dua nilai string
  e. Fungsi *strchr()* untuk mencari nilai karakter dalam string

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1. Ketikkan sebuah kalimat melalui keyboard dengan menggunakan *gets()* (atau *fgets())* kemudian didapatkan keluaran berupa laporan tentang jumlah huruf kecil dan huruf kapital dalam kalimat tsb.

2. Masukkan nama Anda, rubah ke dalam huruf besar semua, balikkan urutan hurufnya, selanjutnya tampilkan hasilnya di layar.

3. Ketikkan sebuah kalimat, hitung dan tampilkan jumlah spasinya.

4. Ketikkan sebuah kalimat, kemudian tampilkan kalimat tsb satu kata perbaris. Asumsikan ada satu spasi yang memisahkan setiap kata dan kalimat diakhiri dengan sebuah tanda titik.

5. Ketikkan sebuah kalimat melalui keyboard kemudian didapatkan keluaran berupa laporan apakah kalimat tsb palindrom ataukah bukan. Misal :

```
Kalimat : KASUR RUSAK
Termasuk PALINDROM

Kalimat : MAKAN MALAM
```

```
Bukan PALINDROM
```

Catatan : disebut palindrom adalah bila urutan kalimat dibalik akan menghasilkan kalimat yang sama. Gunakan berbagai fungsi berkaitan dengan string yang sudah dijelaskan di atas.