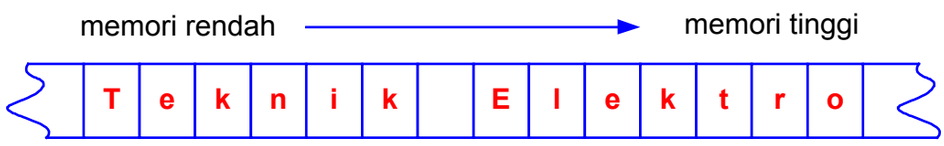
# VIII MANIPULASI STRING

## VIII.1 Pendahuluan

- String merupakan bentuk data yang dapat digunakan untuk menampung dan memanipulasi data teks.
- Dalam bahasa C, string bukan merupakan tipe data tersendiri, namun merupakan jenis khusus dari tipe array.
- Tipe string dapat digunakan sebagai konstanta, yang ditulis dengan diawali dan diakhiri tanda petik ganda. Misalnya:
  " Teknik Elektro"
- Konstanta string seperti di atas disimpan dalam memori secara berurutan, dengan komposisi sebagai berikut:




Gambar 8.1 Struktur penyimpanan string pada memori.

- Setiap karakter akan menempati memori sebesar 1 byte, dan byte terakhir otomatis akan berisi karakter NULL.

## VIII.2 Variabel String

- Variabel string digunakan untuk menyimpan data string. Misalnya

  *char nama[15]*;

- Contoh diatas merupakan instruksi untuk mendeklarasikan variabel tipe string dengan panjang maksimal mengandung 15 karakter (termasuk karakter NuLL)
- Deklarasi di atas sebenarnya adalah deklarasi array bertipe char.
- Untuk memasukkan data string ke dalam suatu variabel dapat dilakukan dengan menggunakan instruksi *gets*( ), dengan bentuk umum pemakaiannya adalah:

  *gets(nama_array)*;

- Jika menggunakan statemen *scanf*( ), maka instruksinya akan menjadi:

  *scanf("%",nama_array)*;

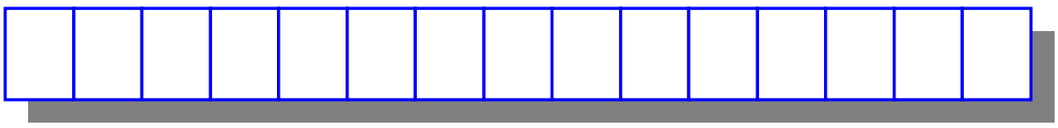
- Didepan nama array tidak perlu ada operator **&**(operator alamat), karena nama array tanpa kurung siku sudah menyatakan alamat.
- Jika menggunakan *scanf*( ), data string masukan tidak bisa mengandung spasi.
- Prototipe *gets*( ) terdapat pada file **stdio.h**.

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

#include <stdio.h>

main()
{
 char nama[15];

 printf("Nama anda: ");
 gets(nama);

 printf("Halo, %s. Selamat datang di Elektro.\n",nama);
}
```

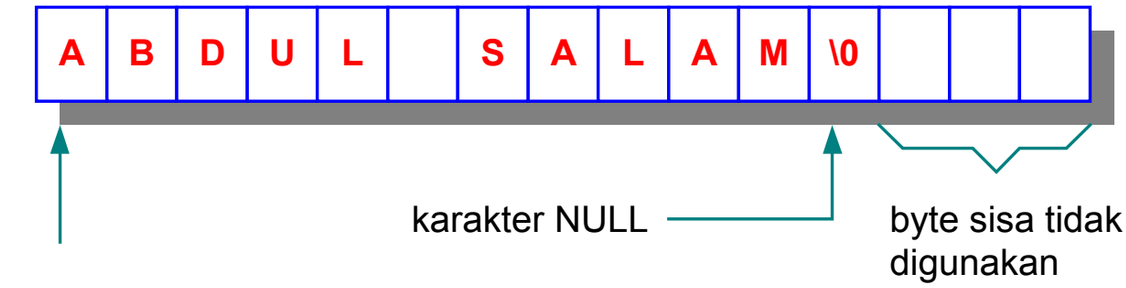
Program 8-1

- Pada program di atas, setelah deklarasi *char nama*[15], maka komputer akan menyediakan ruang pada memori sebanyak 15 karakter.
- Setelah pengguna memasukkan data nama : ABDUL SALAM, maka data string yang dimasukkan akan diletakkan pada area memori yang sudah dipesan.
- Karena data yang dimasukkan kurang dari 15 karakter, maka otomatis setelah karakter terakhir akan diisi dengan karakter NULL.

ruang memori yang disediakan setelah : char nama[15];



Setelah pemasukan data: ABDUL SALAM




Gambar 8.2 Variabel string dan data string.

- Instruksi gets( ) akan membaca seluruh karakter yang diketik pada keyboard sampai tombol ENTER ditekan. Dalam hal ini tidak ada pengecekan terhadap batasan dari array yang merupakan argumennya.
- Jika string yang dimasukkan melebihi ukuran array, maka sisa string (panjang string dikurangi ukuran array plus karakter NULL) akan ditempatkan ke lokasi sesudah bagian akhir array.

## VIII.3 Inisialiasi String

- Suatu variabel string dapat diinisialisasi seperti halnya variabel array yang lain, namun pada elemen terakhir harus berupa karakter NULL. Sebagai contoh:

```
char kota[ ] =
      {'Y', 'o', 'g', 'y', 'a', 'k', 'a', 'r', 't', 'a', '\0'};
```

- Contoh diatas menyatakan bahwa variabel *kota* adalah variabel string dengan nilai awal berupa string "Yogyakarta".
- Bentuk inisialisasi yang lebih singkat adalah:

```
char kota[ ] = "Yogyakarta";
```

- Pada bentuk ini, karakter NULL tidak perlu ditulis, karena secara implisit akan disisipkan oleh kompiler.

## VIII.4 Menampilkan Isi Variabel String Ke Layar

- Untuk menampilkan isi variabel string, dapat digunakan salah satu dari pernyataan berikut:
  - puts(var_string);
  - printf("%s",var_string);
  - printf(var_string);
- Contoh program berikut ini akan menampilkan isi variabel *kota*, berdasarkan dua bentuk inisialisasi string.

```
/* ---------------------------------------------- */
/* File program : kota.c                          */
/* Contoh menampilkan data string ke layar  */
/* ---------------------------------------------- */
#include <stdio.h>
void bentuk1(void);
void bentuk2(void);

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

void bentuk1(void)
{
 char kota[]=
            {'Y','o','g','y','a','k','a','r','t','a','\0'};
 puts(kota);
}

void bentuk2(void)
{
      char kota[] = "Solo";
      puts(kota);
}
```

contoh hasil eksekusi:

```
C>kota
Yogyakarta
Solo
```

---

Program 8-2

## VIII.5 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 mengakses elemen string           */
/* ---------------------------------------- */

#include <stdio.h>
#define maks 256

main()
{
        int i, jumkar;
        char teks[maks];
        puts("Masukkan suatu kalimat.");
        puts("Saya akan menghitung jumlah karakternya.");
        gets(teks);

        jumkar = 0;
        for(i=0;teks[i];i++)
                jumkar++;
        printf("JUmlah karakter = %d\n", jumkar);
}
```

Contoh hasil eksekusi:

```
c>hitkar
Masukkan suatu kalimat.
Saya akan menghitung jumlah karakternya.
Selamat datang
Jumlah karakter = 14
```

---

Program 8-3

- Penghituntgan jumlah karakter dari string *teks* dapat dilakukan dengan memeriksa elemen dari string dimulai dari posisi yang pertama (indeks sama dengan 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 bernilai 0) dilakukan melalui:

```
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 program berikut memperlihatkan cara penyalinan nilai ke suatu variabel string.

---

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

#include <stdio.h>
```

```
main()
{
      int i;
      char keterangan[] = "Saya menyukai bahasa C";
      char kalimat[30];

      i = 0;
      while (keterangan[i] != '\0')
      {
            kalimat[i] = keterangan[i];
            i++;
      }
      kalimat[i] = '\0';      /* Beri karakter NULL */

      printf("Isi kalimat = %s\n", kalimat);
}
```

Contoh hasil eksekusi :

```
c>SALINSTR
Isi kalimat = Saya menyukai bahasa C
```

---

Program 8-4

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

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

- Selama *keterangan*[*i*] tidak berupa karakter NULL, maka *keterangan*[*i*] akan disalin ke *kalimat* [*i*].
- Jadi 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.

- Bentuk yang lebih singkat untuk melakukan penyalinan *keterangan* ke *kalimat* berupa:

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

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

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

## VIII.6 Beberapa Fasilitas Untuk Operasi Karakter

- Turbo C menyediakan sejumlah makro berargumen (semacam fungsi tetapi didefinisikan dengan menggunakan pengarah praprosesor #define) dan fungsi yang berkaian dengan data karakter.
- Fasilitas ini sangat bermanfaat untuk keperluan pemanipulasian string.
- Beberapa diantaranya didefinisikan pada file-judul CTYPE.H, sebagai berikut:

**isalnum()**

- Mempunyai bentuk:

```
int isalnum(int c);
```

- Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf (huruf kapital maupun huruf kecil) atau berupa sebuah karakter angka (0 sampai dengan 9).

**isalpha()**

Mempunyai bentuk:

int isalpha(int c);

- Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf (huruf kapital maupun huruf kecil).

**isdigit()**

- Mempunyai bentuk

  int isdigit(int c);

- Merupakan makro yang menghasilkan nilai benar (bukan nol) jika c adalah sebuah karakter angka (0 sampai dengan 9).

**islower()**

- Mempunyai bentuk

  int islower(int c);

- Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf kecil (‘a’ sampai dengan ‘z’).

**isupper()**

- Mempunyai bentuk

  int isupper(int c);

- Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf kecil (‘A’ sampai dengan ‘Z’).

**tolower()**

- Mempunyai bentuk

  int tolower(int c);

- Jika c adalah huruf kapital, maka hasil fungsi berupa huruf kecilnya.
- Apabila c tidak berupa huruf kapital, keluaran fungsi sama dengan c.

**toupper()**

- Mempunyai bentuk

```
int toupper(int c);
```

- Jika c adalah huruf kecil, maka hasil fungsi berupa huruf kapitalnya.
- Apabila c tidak berupa huruf kecil, keluaran fungsi sama dengan c.

**strcpy()**

- Mempunyai bentuk

```
strcpy(tujuan, asal);
```

- Fungsi ini digunakan untuk menyalin variabel string *asal* ke variabel string *tujuan*.
- Dalam hal ini, variabel *tujuan* haruslah mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string *asal*.
- Contoh:

```
strcpy(pepatah, "Kalau ada kemauan pasti ada jalan");
```

- merupakan instruksi untuk menyalin string "*Kalau ada kemauan pasti ada jalan*" ke variabel string *pepatah*.

**strlen()**

- Mempunyai bentuk

```
strlen(var_string);
```

- Digunakan untuk memperoleh jumlah karakter dalam variabel string yang merupakan argumennya.

- Karakter NULL tidak ikut dihitung.

**strcat()**

- Mempunyai bentuk

```
strcat(tujuan, sumber);
```

- Fungsi ini digunakan untuk menambahkan variabel string *sumber* ke bagian akhir dari variabel string *tujuan*.

**strcmp()**

- Mempunyai bentuk

```
var_int = strcmp(str1, str2);
```

- Fungsi ini digunakan untuk membandingkan variabel string *str1* dengan string *str2*.
- Hasil fungsi bertipe *int* berupa nilai
  - Negatif, jika *str1* kurang dari *str2*
  - Nol, jika *str1* sama dengan *str2*
  - Positif, jika *str1* lebih dari *str2*
- Nilai absolut hasil fungsi (kecuali jika bernilai nol) menyatakan selisih nilai ASCII dari karakter yang menyebabkan *str1* berbeda dengan *str2*.
- Pembandingan dilakukan untuk karater 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 dari karakter ‘B’ dan karakter ‘B’ lebih kecil dari karakter ‘C’.
- Contoh : string “HALO” lebih kecil dari string “HELO”, karena karakter ‘A’ mempunyai nilai yang lebih kecil daripada karakter ‘E’.
- Apabila salah satu string mempunyai panjang yang lebih pendek, dan sampai karakter yang terakhir dari string yang terpendek ternyata karakter antara *str1* dengan *str2* sama, maka string yang lebih pendek mempunyai nilai yang lebih kecil dibandingkan dengan string yang lebih panjang.

**Latihan**

Berdasarkan string “BORLAND”, buatlah program yang akan menghasilkan keluaran sebagai berikut:

D
ND
AND
LAND
RLAND
ORLAND
BORLAND