

Bab 10. String

Konsep Pemrograman
Politeknik Elektronika Negeri Surabaya
2006

Overview

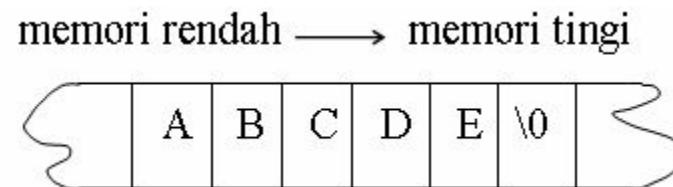
- Pendahuluan
- Konstanta String
- Variabel String
- Inisialisasi String
- Input Output Data String
 - Memasukkan Data String
 - Menampilkan Data String
- Mengakses Elemen String
- *Built-in Functions* untuk manipulasi String
 - Fungsi `strcpy()`
 - Fungsi `strlen()`
 - Fungsi `strrev()`
 - Fungsi `strcmp()`
 - Fungsi `strcmpi()`

Pendahuluan

- 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 → *array of char*.

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:



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

Variabel String

- Variabel string adalah variabel yang dipakai untuk 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*.

Inisialisasi String

- Suatu variabel string dapat diinisialisasi seperti halnya array yang lain (dalam kurung kurawal dipisahkan koma). 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.



Input Output Data String

- Untuk memasukkan atau menampilkan data String digunakan bisa beberapa fungsi standar yang ada di `stdio.h`.
- Untuk operasi input :
 - `scanf()`
 - `gets()`
 - `fgets()`
- Untuk operasi output :
 - `puts()`
 - `printf()`

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);
```




Memasukkan Data String

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

Memasukkan Data String

```
main() {  
    char name[15];  
  
    printf("Masukkan nama Anda : ");  
    gets(name);  
    printf("\nHalo, %s. Selamat belajar string.\n", name);  
}
```

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

name :

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Setelah data yang dimasukkan berupa : **SAIFUDDIN**

name :

S	A	I	F	U	D	D	I	N	\0					
---	---	---	---	---	---	---	---	---	----	--	--	--	--	--

↑
karakter NULL

byte sisa tak dipakai

Memasukkan Data String

- Perlu diketahui, fungsi `gets()` akan membaca seluruh karakter yang diketik melalui keyboard sampai tombol ENTER ditekan dengan tanpa mengecek 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);
```

Uji Coba dengan `gets()`

```
#include <stdio.h>
#define MAKS 5
```

```
main() {
```

```
    char kar = 'A';
```

```
    char nama[MAKS];
```

```
    printf("Karakternya = %c\n", kar);
```

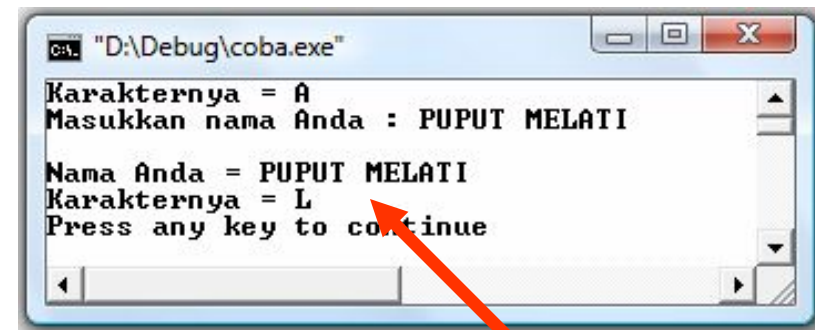
```
    printf("Masukkan nama Anda : ");
```

```
    gets(nama);
```

```
    printf("\nNama Anda = %s\n", nama);
```

```
    printf("Karakternya = %c\n", kar);
```

```
}
```



Input string melebihi kapasitas array sehingga menumpuki data tetangganya

Menampilkan Data 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);
```

Mengakses Elemen String

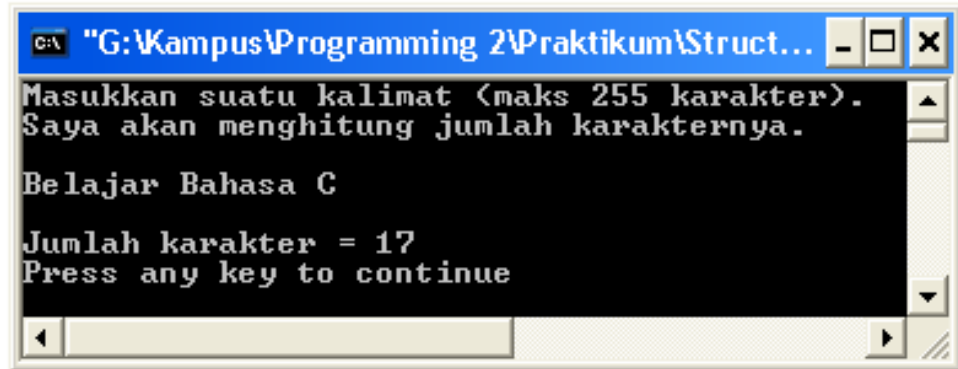
- Variabel string merupakan bentuk khusus dari array bertipe *char*. Oleh karena itu, elemen dari variabel string dapat diakses seperti halnya pengaksesan elemen pada array.
- Perhitungan jumlah karakter dari string teks dapat dilakukan dengan memeriksa elemen dari string dimulai dari posisi yang pertama (indeks ke-0) sampai ditemukannya karakter NULL.
- Program berikut menunjukkan cara mengakses elemen array untuk menghitung total karakter dari string yang dimasukkan melalui keyboard

Mengakses Elemen String

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



```
C:\ "G:\Kampus\Programming 2\Praktikum\Struct..."
Masukkan suatu kalimat (maks 255 karakter).
Saya akan menghitung jumlah karakternya.

Belajar Bahasa C

Jumlah karakter = 17
Press any key to continue
```

Built-in Functions untuk manipulasi String

- Untuk manipulasi string, C telah menyediakan beberapa fungsi standar yang ada pada `string.h`
- Beberapa yang akan dibahas kali ini adalah
 - Fungsi `strcpy()`
 - Fungsi `strlen()`
 - Fungsi `strrev()`
 - Fungsi `strcmp()`
 - Fungsi `strcmpi()`

Fungsi `strcpy()`

- Bentuk pemakaian :

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

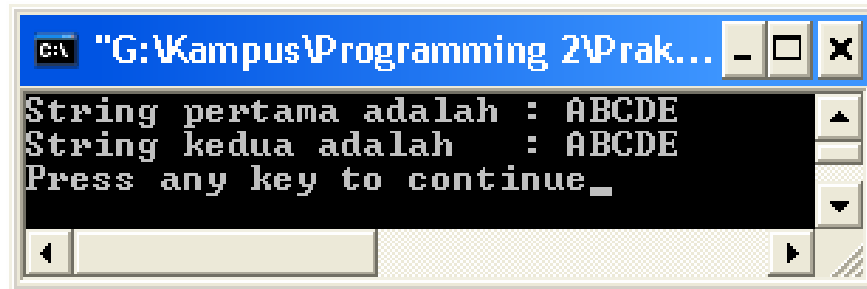
- Fungsi ini dipakai untuk mengcopy string `asal` ke variabel string `tujuan` termasuk karakter `'\0'`.
- Dalam hal ini, variabel `tujuan` haruslah mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string `asal`

Fungsi strcpy()

```
#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);
}
```



Fungsi `strlen()`

- Bentuk pemakaian :

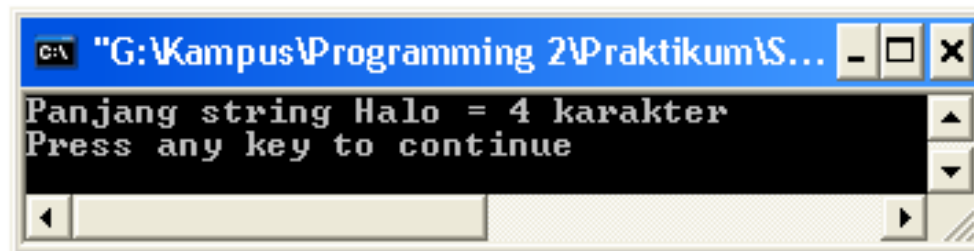
```
#include <string.h>
strlen(var_string);
```
- Fungsi ini digunakan untuk memperoleh banyaknya karakter di dalam string yang menjadi argumennya (`var_string`).
- Keluaran dari fungsi ini adalah panjang dari `var_string` (karakter NULL tidak ikut dihitung)

Fungsi strlen()

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

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

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



```
C:\ "G:\Kampus\Programming 2\Praktikum\S...
Panjang string Halo = 4 karakter
Press any key to continue
```

Fungsi `strcmp()` (*case sensitive*)

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

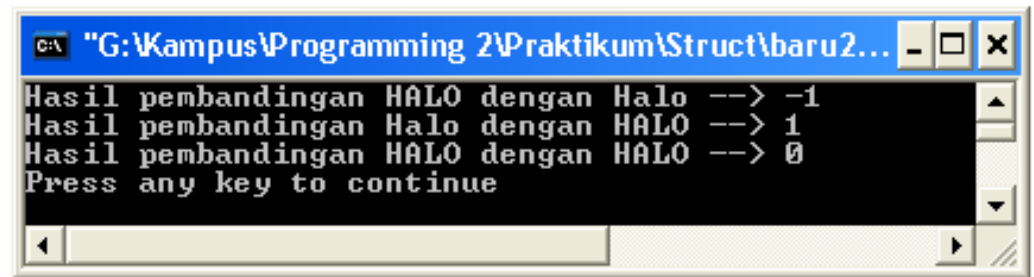
- Contoh bentuk pemakaian fungsi :

```
#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`
- Perbandingan dilakukan untuk karakter pada posisi yang sama dari `str1` dan `str2`, dimulai dari karakter terkiri yang didasarkan oleh nilai ASCII-nya. Misal, karakter 'A' lebih kecil daripada 'B' dan karakter 'B' lebih kecil daripada 'C'.

Fungsi strcmp ()

```
#include <stdio.h>
#include <string.h>
main() {
    char str1[]="HALO";
    char str2[]="Halo";
    char str3[]="HALO";
```



```
"G:\Kampus\Programming 2\Praktikum\Struct\baru2..."
Hasil pembandingan HALO dengan Halo --> -1
Hasil pembandingan Halo dengan HALO --> 1
Hasil pembandingan HALO dengan HALO --> 0
Press any key to continue
```

```
    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));
}
```

Fungsi `strcmpi()` (*non case sensitive*)

- Membandingkan dua buah nilai string secara *non case sensitive* dapat dilakukan dengan fungsi `strcmpi()`.
- Contoh bentuk pemakaian fungsi :

```
#include <string.h>
strcmpi(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`

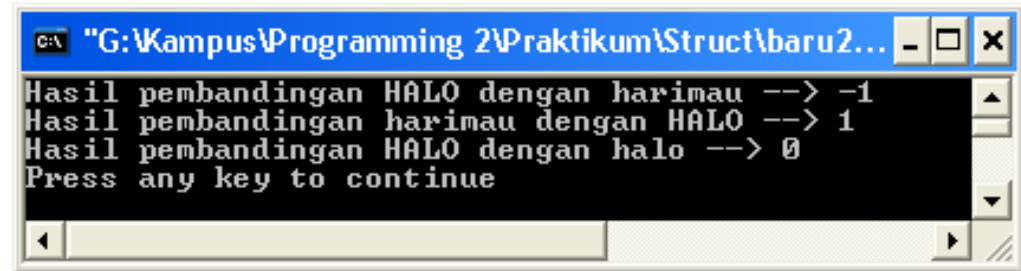
Fungsi strcmpi ()

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

```
main() {
    char str1[]="HALO";
    char str2[]="harimau";
    char str3[]="halo";
```

```
    printf("Hasil pembandingan %s dengan %s --> %d\n",
           str1, str2, strcmpi(str1, str2));
    printf("Hasil pembandingan %s dengan %s --> %d\n",
           str2, str1, strcmpi(str2, str1));
    printf("Hasil pembandingan %s dengan %s --> %d\n",
           str1, str3, strcmpi(str1, str3));
```

```
}
```



Latihan

- Ketikkan semua contoh program yang ada pada modul teori (10.String.ppt)
- Running setiap program dan amatilah outputnya
- Berikan analisis dan kesimpulan pada setiap contoh program tsb

Latihan

String handling → *User defined function*

1. Lakukan percobaan untuk menginputkan string dari keyboard dengan menggunakan : `scanf()` , `gets()` dan `fgets()` . Analisislah dan berikan kesimpulan untuk setiap fungsi tsb.
2. Buatlah program untuk menerima input string dari keyboard kemudian hitunglah panjang dari string tsb dan tampilkan hasilnya
3. Lanjutkan program nomor 2 untuk membalik string tsb, misalnya : budi → ibud
4. Buatlah program yang mendeklarasikan sekaligus menginisialisasi sebuah array `kata1[]` , kemudian copy-lah isi array `kata1[]` tsb ke dalam array `kata2[]` , selanjutnya tampilkan isi kedua array tsb ke layar

Latihan

String Handling -> *built in functions*

5. Ulangilah soal nomor 2, 3 & 4 di atas dengan menggunakan fungsi-fungsi standard
6. Lakukan percobaan untuk membandingkan 2 buah string dengan menggunakan fungsi `strcmp()` dan `strcmpi()`. Analisislah dan berikan kesimpulan tentang perbedaan dan contoh aplikasi untuk keduanya.