TIF1101 – Dasar-Dasar Pemrograman

HO 15 - String

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Pendahuluan

- Array karakter membentuk satu kelas array khusus yang disebut string.
- Contoh:
  - Konstanta string, adalah sebuah array satu-dimensi berjenis karakter, yang diakhiri dengan karakter nol (string sentinel), ”\0”.
  - Array berjenis **char** juga dapat diperlakukan sebagai sebuah jenis data string yang variabel-variabelnya dapat menampung konstanta string.

Mendeklarasi Variabel String

- String dapat dideklarasian dengan dua cara.
  1. Pendeklarasian sebagai array karakter:
     char nama_string[ukuran];
     - Contoh:
       char alamat [30];
       char kota [15];
     - Agar variabel-variabel tersebut diperlakukan sebagai string, maka elemen terakhir dari array tersebut harus disisihkan untuk menampung karakter nol.

Mendeklarasi Variabel String

2. Menggunakan pointer ke karakter:

- Contoh:

  char *alamat, *kota;
- Penting!
  - Alokasikan memori yang digunakan untuk menampung string.
  - Contoh:
    alamat = (char *) malloc(30);
    kota = (char *) malloc(15);

Menginisialisasi Variabel String

- Inisialisasi string pada saat deklarasi dapat dilakukan dengan dua cara.
  1. Tuliskan karakter-karakternya satu persatu dan tambahkan karakter nol pada elemen terakhir.
     - Contoh:
       char kota [8] = {'B','A','N','D','U','N','G','\0'};
       char kota [ ] = {'M','E','D','A','N','\0'};
  2. Berikan konstanta string.
     - Contoh:
       char kota [ ] = "CORVALLIS";

Menginisialisasi Variabel String

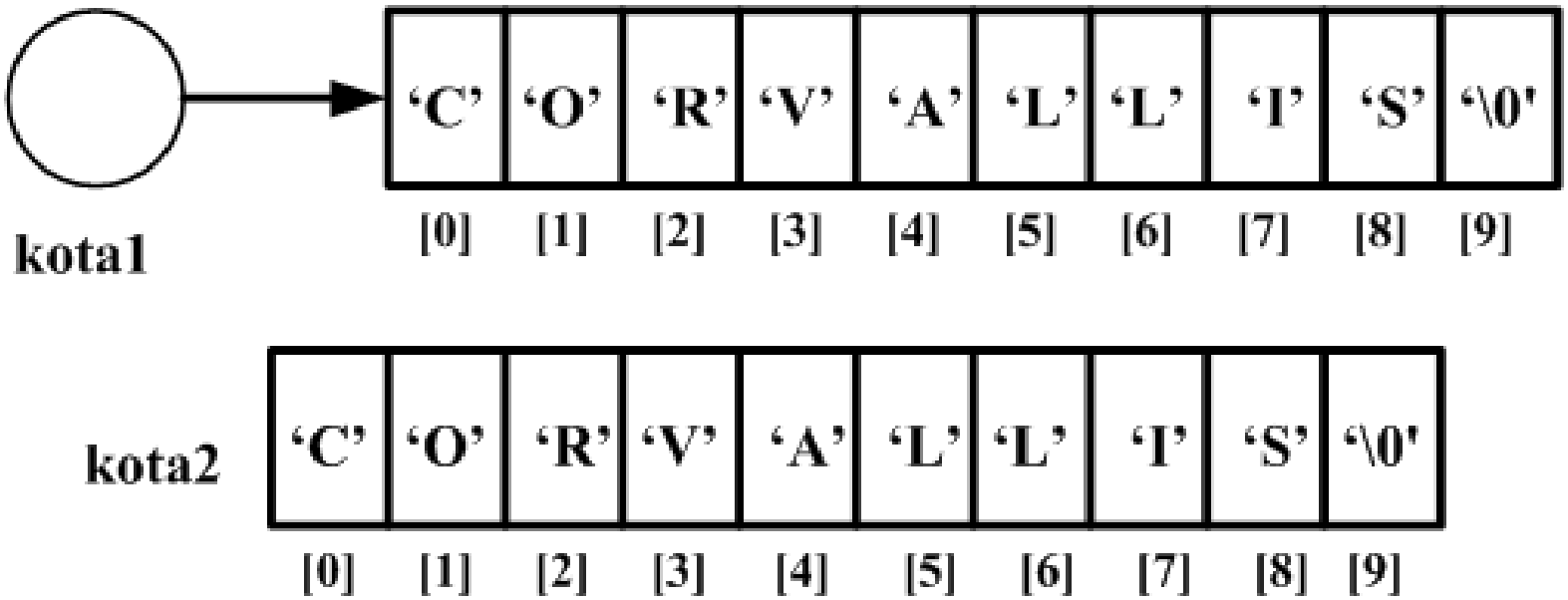
- Jika variabel string dideklarasi menggunakan pointer, inisialisasi variabel hanya dapat dilakukan dengan memberikan konstanta string pada saat deklarasi .
- Contoh:
  char *kota = "CORVALLIS";
- Tampilkan di layar:
  printf("%s, %c%c\n", kota, kota [1], kota [2]);
- Output:
  CORVALLIS, OR

Menginisialisasi Variabel String

- Meskipun array dan pointer memiliki penggunaan yang sama, pada prinsipnya terdapat beberapa perbedaan.
- Contoh:
  char *kota1 = "CORVALLIS";
  char kota2[ ] = "CORVALLIS";
- Pada pendeklarasian pertama:
  1. Kompiler mengalokasikan memori untuk *kota1*
  2. Meletakkan konstanta string "CORVALLIS" dalam memori
  3. Memberikan pada *kota1* alamat dasar dari konstanta string tersebut, sehingga, variabel *kota1* menunjuk ke string.
  4. String C yang diinisialisasi melalui sebuah pointer ke karakter tidak dapat diubah.
  5. Upaya untuk mengubah string tersebut menyebabkan **undefined error**.

Menginisialisasi Variabel String

- Keadaan memori di atas dapat diperlihatkan seperti pada gambar.



Menginisialisasi Variabel String

```
1  /* Nama file: Contoh13_1.c
2     Deklarasi pointer untuk string */
3  #include <stdio.h>
4
5  int main()
6  {
7     char *kota1 = "Corvallis", temp;
8
9     /* temp = kota1[2];
10    kota1[2] = kota1[1];
11    kota1[1] = temp; */
12
13    printf("%s\n", kota1);
14
15    return 0;
16 }
```

Menginisialisasi Variabel String

```
1  /* Namafile: cthStringPtr1.c
2     Alamat memori untuk string pointer */
3  #include <stdio.h>
4
5  int main()
6  {
7     char *kota1 = "CORVALLIS";
8     int i;
9
10    printf ("%s, %c%c\n", kota1, *(kota1+1), *(
          kota1+2));
```

Menginisialisasi Variabel String

```
11    printf("Alamat memori kota1=%u\n", &kota1);
12    printf("Isi alamat %u adalah %u\n", &kota1,
          &kota1[0]);
13    for(i=0; i<9; i++)
14      printf("Alamat memori kota1[%d] = %u\n", i
            , &kota1[i]);
15    return 0;
16  }
```

Menginisialisasi Variabel String

- Pada pendeklarasian kedua:
  1. Kompiler akan menghitung jumlah memori yang harus dialokasikan pada array *kota2*, dalam hal ini adalah sebesar 10 byte.
  2. String C yang diinisialisasi melalui deklarasi karakter array dapat diubah.
  3. Contoh:
     temp = kota2[2];
     kota2[2] = kota2[1];
     kota2[1] = temp;

Menginisialisasi Variabel String

```
1  /* Nama file: Contoh13_2.c
2     Deklarasi array untuk string */
3  #include <stdio.h>
4
5  int main()
6  {
7     char kota2[] = "Corvallis", temp;
8
9     temp = kota2[2];
10    kota2[2] = kota2[1];
11    kota2[1] = temp;
12
13    printf("%s\n", kota2);
14
15    return 0;
16 }
```

Menginisialisasi Variabel String

```
1  /* Namafile: cthStringPtr2.c
2     Alamat memori array karakter */
3  #include <stdio.h>
4
5  int main()
6  {
7     char kota1[] = "CORVALLIS";
8     int i;
9
10    printf("%s, %c%c\n", kota1, kota1[1], kota1
          [2]);
```

Menginisialisasi Variabel String

```
11    printf("Alamat memori kota1 = %u\n", &kota1)
          ;
12    printf("Isi alamat %u adalah %u\n", &kota1,
          &kota1[0]);
13    for(i=0; i<9; i++)
14      printf("Alamat memori kota1[%d] = %u\n", i
            , &kota1[i]);
15
16    return 0;
17  }
```

Mengalokasikan Memory

- Apabila variabel pointer mendapatkan inisialisasi berupa string pada saat dideklarasikan, kompiler otomatis mengalokasikan memori untuk string yang ditunjuk oleh variabel tersebut.
- Akan tetapi, apabila variabel pointer ke string tidak segera diberi nilai pada saat deklarasi, tetapi dilakukan di bagian pernyataan yang dapat dieksekusi.
- Variabel pointer ke string tersebut harus terlebih dahulu dialokasikan memori sebelum dapat menyimpan sebuah nilai string tertentu.

Mengalokasikan Memory

```
1  /* Namafile: cthStringPtr3.c
2     Alokasi dinamik untuk string */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #define KOTA "CORVALLIS"
7
8  int main()
9  {
10    char *kota1;
11    int i;
12
13    kota1 = (char *) malloc(sizeof(KOTA));
14
15    strcpy(kota1, KOTA);
```

Mengalokasikan Memory

```
16    printf ("Alokasi memori=%u\n", sizeof(char)*
          strlen(KOTA)+1);
17    printf ("%s, %c%c\n", kota1, *(kota1+1), *(
          kota1+2));
18    printf("Alamat memori kota1=%u\n", &kota1);
19    printf("Isi alamat %u adalah %u\n", &kota1,
          &kota1[0]);
20    for(i=0; i<9; i++)
21      printf("Alamat memori kota1[%d] = %u\n", i
            ,&kota1[i]);
22
23    return 0;
24  }
```

Fungsi-Fungsi Pengolahan String

- Fungsi-fungsi standar penanganan string yang prototipenya terdapat pada berkas judul *string.h*.
- Untuk menggunakan fungsi-fungsi standar tersebut berkas judul ini harus disertakan menggunakan prapengolah #include.

Menyalin string (*String copy*)

- Memberikan sebuah string pada suatu variabel.
- Menerima dua buah argumen berjenis string dan mengembalikan sebuah pointer ke karakter.
- Prototipe Fungsi:
  char *strcpy(**char** *str1, **const char** *str2);
- Konstanta string *str2* disalinkan ke variabel string *str1*.
- Contoh:
  strcpy(kota, "CORVALLIS");

Menyalin string (*String copy*)

```
1  /* Namafile: cthstrcpy.c
2     Menyalin string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char kata1[] = "Selamat";
9      char kata2[11];
10
11     strcpy(kata2, kata1);
12     printf("Kata 1 adalah: \"%s\"\n", kata1);
13     printf("Kata 2 adalah: \"%s\"\n", kata2);
14
15     return 0;
16 }
```

Menyalin string (*String copy*)

- Sebuah variasi fungsi penyalinan string lain, yaitu **strncpy()** digunakan untuk menyalin hingga ke karakter tertentu yang ditunjuk oleh *str2* ke dalam string yang ditunjuk oleh *str1*.
- Prototipe fungsi **strncpy()** adalah:

  char *strncpy(**char** *str1, **const char** *str2, **size_t** count);

Menyalin string (*String copy*)

```
1  /* Namafile: cthstrncpy.c
2     Menyalin string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     char kata1[ ] = "Selamat";
9     char kata2[11];
10
11    strncpy(kata2, kata1, 5);
12    kata2[5] = '\0';
```

Menyalin string (*String copy*)

```
13    printf("Kata 1 adalah: \"%s\"\n", kata1);
14    printf("Kata 2 adalah: \"%s\"\n", kata2);
15
16    return 0;
17  }
```

Menyambung string (*String concatenation*)

- Dua buah string dapat disambung ke sebuah string lain dengan menggunakan fungsi standar **strcat**().
- Prototipe fungsi **strcat**():
  char *strcat(**char** *str1, **const char** *str2);
- Contoh:
  char kata1[14] = "Kapal ";
  char kata2[ ] = "Selamat";

  strcat (kata1, kata2);

Menyambung string (*String concatenation*)

```
1  /*  Namafile: cthstrcat.c
2    Menyalin string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8    char kata1[14] = "Kapal ";
9    char kata2[ ] = "Selamat";
10
11   printf("Kata 1 adalah: \"%s\"\n", kata1);
12   printf("Kata 2 adalah: \"%s\"\n", kata2);
```

Menyambung string (*String concatenation*)

```
13    strcat (kata1, kata2);
14
15    printf("Kata 1 adalah: \"%s\"\n", kata1);
16    printf("Kata 2 adalah: \"%s\"\n", kata2);
17
18    return 0;
19  }
```

Menyambung string (*String concatenation*)

- Variasi fungsi penyalinan string lain, yaitu **strncpy()** untuk menyalin hingga ke karakter tertentu yang ditunjuk oleh **str2** ke dalam string yang ditunjuk oleh **str1**.
- Prototipe fungsi strncpy() adalah:

  strncpy(**char** *str1, **const char** *str2, **size_t** count);
- Contoh:
  char kata1[12] = "Kapal ";
  char kata2[ ] = "Selamat";

  strncat(kata1, kata2, 5);

Menyambung string (*String concatenation*)

```
1  /*  Namafile: cthstrncat.c
2    Menyalin string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8    char kata1[14] = "Kapal ";
9    char kata2[ ] = "Selamat";
10
11   printf("Kata 1 adalah: \"%s\"\n", kata1);
12   printf("Kata 2 adalah: \"%s\"\n", kata2);
```

Menyambung string (*String concatenation*)

```
13    strncat (kata1, kata2);
14
15    printf("Kata 1 adalah: \"%s\"\n", kata1);
16    printf("Kata 2 adalah: \"%s\"\n", kata2);
17
18    return 0;
19  }
```

Menghitung panjang string (*String length*)

- Fungsi **strlen()** akan mengembalikan panjang string yang diakhiri dengan karakter nol, tidak termasuk karakter nol itu sendiri.
- Prototipe fungsi strlen() adalah:

  size_t strlen (**char** *str);
- Contoh:
  char kata1[ ] = "Kapal "; char kata2[ ] = "Selamat";
  strlen(kata1); strlen(kata2);

Menghitung panjang string (*String length*)

```
1  /*  Namafile: cthstrlen.c
2    Menghitung panjang string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8    char kata1[14] = "Kapal ";
9    char kata2[ ] = "Selamat";
10
11   printf("Kata 1: \"%s\", panjang=%d\n", kata1
         , strlen(kata1));
12   printf("Kata 2: \"%s\", panjang=%d\n", kata2
         , strlen(kata2));
```

Menghitung panjang string (*String length*)

```
13    strncat (kata1, kata2, 5);
14    printf("Kata 1: \"%s\", panjang=%d\n", kata1
          , strlen(kata1));
15    printf("Kata 2: \"%s\", panjang=%d\n", kata2
          , strlen(kata2));
16
17    return 0;
18  }
```

Membandingkan string (*String compare*)

- Fungsi **strcmp()** secara alfabetis membandingkan dua buah string dan mengembalikan sebuah nilai integer berdasarkan hasil yang diperoleh, yaitu:
  - Lebih kecil dari 0, jika string pertama lebih kecil daripada string kedua.
  - 0, jika string pertama sama dengan string kedua.
  - Lebih besar dari 0, jika string pertama lebih besar daripada string kedua.
- Prototipe fungsi strcmp() adalah:

  int strcmp(**const char** *str1, **const char** str2);

Membandingkan string (*String compare*)

- Contoh:

```
char kata1[ ] = "Kapal ";
char kata2[ ] = "Selamat";
char kata3[7];

strcmp(kata1, kata2);
strcpy(kata3, kata1);
strcmp(kata1, kata3);
strcmp(kata2, kata3);
```

Membandingkan string (*String compare*)

```
1  /*  Namafile: cthstrcmp.c
2    Membandingkan string */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8      char kata1[ ] = "Kapal ";
9      char kata2[ ] = "Selamat";
10     char kata3[7];
11
12     printf("Kata 1: \"%s\"\n", kata1);
13     printf("Kata 2: \"%s\"\n", kata2);
```

Membandingkan string (*String compare*)

```
14    printf("Hasil perbandingan %s dengan %s = %d
          \n", kata1, kata2, strcmp(kata1, kata2));
15    strcpy(kata3, kata1);
16    printf("Hasil perbandingan %s dengan %s = %d
          \n", kata1, kata3, strcmp(kata1, kata3));
17    printf("Hasil perbandingan %s dengan %s = %d
          \n", kata2, kata3, strcmp(kata2, kata3));
18
19      return 0;
20  }
```

Membandingkan string (*String compare*)

- Pembadingan string juga dapat dilakukan hingga sejumlah karakter tertentu, yaitu menggunakan fungsi strncmp().
- Prototipe fungsi ini adalah sebagai berikut: **int**

  strncmp(**const char** *str1, **const char** str2, size_t count);
- Contoh:
  char nim1[ ] = "191402001";
  char nim2[ ] = "191402100";

  strncmp(nim1, nim2, 7);
  strncmp(nim1, nim2, 6);
  strncmp(nim2, nim1, 7);