

BLM 112- Programlama Dilleri II

Hafta 7

String Fonksiyonları Sıralama & Arama

Dr. Öğr. Üyesi Caner Özcan

Bilim doğanın dilini anlama çabasıdır. O dili anlayan doğayı dost, anlamayan düşman bilir.

String

String NULL karakter '\0' ile biten bir karakter dizisidir.

Örnek: `char str[8];`

- ▶ Ençok 8 karakter alabilen bir dizi oluşturur.
- ▶ Eğer str dizisi string olarak kullanılacak ise en fazla 7 karakter alabilir ve sonu NULL karakter '\0' ile bitmek zorundadır.



String işlemleri

- ▶ C standard kütüphanesi stringleri manipüle etmek için birçok fonksiyon içerir.
- ▶ Bu fonksiyonları kullanmak için `<string.h>` ifadesini eklemeniz gerekir.

`#include <string.h>`

- ▶ Bazı önemli fonksiyonlar:
 - **`strcpy(char *str1, const char *str2);`**
 - **`strlen(const char *str);`**
 - **`strcat(char *str1, const char *str2);`**
 - **`strcmp(const char *str1, const char *str2);`**

Strcpy Fonksiyonu

```
char *strcpy(char *str1, const char *str2)
{
    char *p = str1;

    while (*str2)
        *p++ = *str2++;

    *p = '\0';
    return str1;
} /* end-strcpy */
```



Strcpy Fonksiyonu

```
#include <stdio.h>
#include <string.h>
int main () {
    char src[40];
    char dest[100];
    strcpy(src, "This is tutorialspoint.com");
    strcpy(dest, src);
    printf("Final copied string : %s\n", dest);
    return(0);
}
```

Output:

Final copied string : This is tutorialspoint.com



• Strlen Fonksiyonu

```
int strlen(const char *str)
{
    int len = 0;

    while(*str++)
        len++;

    return len;
} /* end-strlen */
```



Strlen Fonksiyonu

```
#include <stdio.h>
#include <string.h>
int main () {
    char str[50];
    int len;
    strcpy(str, "This is tutorialspoint.com");
    len = strlen(str);
    printf("Length of |%s| is |%d|\n", str, len);
    return(0);
}
```

Output:

Length of |This is tutorialspoint.com| is |26|



Strcat Fonksiyonu

```
char *strcat(char *str1, const char *str2)
{
    char *p = str1;

    while(*p)
        p++;

    while(*str2)
        *p++ = *str2++;

    *p = '\0';
    return str1;
} /* end-strcat */
```


Strcat Fonksiyonu

```
#include <stdio.h>
#include <string.h>
int main () {
    char src[50],
    dest[50];
    strcpy(src, "This is source");
    strcpy(dest, "This is destination");
    strcat(dest, src);
    printf("Final destination string : |%s|", dest);
    return(0);
}
```

Output:

Final destination string : |This is destinationThis is source|



Strcmp Fonksiyonu

```
int strcmp(const char *str1, const char *str2)
{
    while (*str1 && *str2 && *str1 == *str2){
        str1++; str2++;
    } /* end-while */

    return *str1-*str2;
} /* end-strcmp */
```



Strcmp Fonksiyonu

```
#include <stdio.h>
#include <string.h>
int main () {
    char str1[15], str2[15];
    int ret;
    strcpy(str1, "abcdef");
    strcpy(str2, "ABCDEF");
    ret = strcmp(str1, str2);
    if(ret < 0)
        printf("str1 is less than str2");
    else if(ret > 0)
        printf("str2 is less than str1");
    else
        printf("str1 is equal to str2");
    return(0);
}
```

Output:

str2 is less than str1



Sıralama

- ▶ Bir grup veriyi azalan veya artan şekilde yerleştirme.
- ▶ Bilgisayar sistemleri için veri sıralama çok önemlidir.
- ▶ Sıralama işlemi, hem arama işlemlerini hem de bir grup veriyi listeleme işlemini hızlandırmaya yarar.
- ▶ En popüler sıralama algoritmaları:
 - Insertion sort (Araya ekleme sıralaması)
 - Selection sort (Seçim sıralaması)
 - Bubble sort (Kabarcık sıralaması)
 - Quick sort (Hızlı sıralama)



Kabarcık Sıralaması (Bubble Sort)

- ▶ Zaman karmaşıklığı $O(n^2)$
- ▶ Eğer n adet elemandan c adet eleman sıralı değilse zaman karmaşıklığı $O(c n)$
- ▶ Algoritmanın tasarımı kolaydır ancak algoritma verimli değildir.
- ▶ Az sayıda eleman üzerinde ya da çoğu elemanı zaten sıralanmış listeler üzerinde kullanılabilir.



Kabarcık Sıralaması (Bubble Sort)

Sıralanacak dizi: [7,3,5,1,2]

- Hareket 1- Çevrim-1
[3,7,5,1,2]
- Hareket 1- Çevrim-2
[3,5,7,1,2]
- Hareket 1- Çevrim-3
[3,5,1,7,2]
- Hareket 1- Çevrim-4
[3,5,1,2,7]

- Hareket 2- Çevrim-1
[3,5,1,2,7]
- Hareket 2- Çevrim-2
[3,1,5,2,7]
- Hareket 2- Çevrim-3
[3,1,2,5,7]
- Hareket 2- Çevrim-4
[3,1,2,5,7]

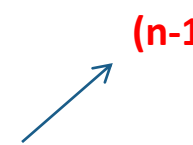
- Hareket 3- Çevrim-1
[1,3,2,5,7]
- Hareket 3- Çevrim-2
[1,2,3,5,7]
- Hareket 3- Çevrim-3
[1,3,2,5,7]
- Hareket 3- Çevrim-4
[1,3,2,5,7]

- Hareket 4- Çevrim-1
[1,3,2,5,7]
- Hareket 4- Çevrim-2
[1,2,3,5,7]
- Hareket 4- Çevrim-3
[1,2,3,5,7]
- Hareket 4- Çevrim-4
[1,2,3,5,7]



Kabarcık Sıralaması (Bubble Sort)

```
26 void bubbleSort(int dizi[],int n)
27 {
28     int gecici;
29
30     for (int i = 0; i < n; i++)
31     {
32         for (int k = 0; k < n - 1 - i; k++)
33         {
34             if(dizi[k]>dizi[k+1])
35             {
36                 gecici=dizi[k];
37                 dizi[k] = dizi[k + 1];
38                 dizi[k + 1] = gecici;
39             }
40         }
41     }
42 }
```



Kabarcık Sıralaması (Bubble Sort)

```
1 #include <stdio.h>
2
3 void bubbleSort(int [],int);
4 int main(void)
5 {
6     int i=0,a[5];
7     printf("Sıralamak istediğin 5 sayi gir\n");
8     while(i<5){
9         scanf("%d",&a[i]);
10        i++;
11    }
12    i=0;
13    bubbleSort(a,5);
14
15    printf("Bubble sort isleminden sonra...\n");
16    while(i<5){
17        printf("%d ",a[i]);
18        i++;
19    }
20    return 0;
21 }
```


Araya Ekleme (Insertion Sort)

- ▶ Sıralı bir listeye eleman eklemek için uygundur.
- ▶ Sıralı bir listeye eleman eklemenin karmaşıklığı : $O(n)$
- ▶ Eğer liste veya dizi sıralı değilse karmaşıklığı : $O(n^2)$



Araya Ekleme (Insertion Sort)

Sıralanacak dizi: [7,3,5,8,2]

Başlangıç durumu : [7][3,5,8,2]

Önce	Sonra
[7, 3] [5, 8, 2]	[3, 7] [5, 8, 2]
[3, 7, 5] [8, 2]	[3, 5, 7] [8, 2]
[3, 5, 7, 8] [2]	[3, 5, 7, 8] [2]
[3, 5, 7, 8, 2] []	[2, 3, 5, 7, 8] []

Araya Ekleme (Insertion Sort)

```
23 void insertionSort(int dizi[], int n)
24 {
25     int ekle,k,i;
26     for (i = 1; i < n; i++)
27     {
28         ekle = dizi[i];
29         for (k = i - 1; k >= 0 && ekle <= dizi[k]; k--)
30             dizi[k + 1] = dizi[k]; //Geriye kaydırma
31         dizi[k + 1] = ekle;         //Uygun yer boşaltıldı eklendi
32     }
33 }
```

Araya Ekleme (Insertion Sort)

```
1 #include <stdio.h>
2
3 void insertionSort(int [],int);
4 int main(void)
5 {
6     int i=0,a[5];
7     printf("Siralamak istediğın 5 sayı gir\n");
8     while(i<5){
9         scanf("%d",&a[i]);
10        i++;
11    }
12    i=0;
13    insertionSort(a,5);
14
15    printf("Insertion sort isleminde sonra...\n");
16    while(i<5){
17        printf("%d ",a[i]);
18        i++;
19    }
20    return 0;
21 }
```

Seim Sıralaması (Selection Sort)

- ▶ Eğer bir eleman olması gereken yerde ise bulunduğu sıra değıştirilmez.
- ▶ Yarı yarıya sıralanmış bir grup veri olması durumunda eleman yeri değışimi azdır.
- ▶ Listedeki ilk elemanı al ve bu elemanı diğer elemanlar arasından en küçüğü ile değıştir. Bu işlemi son elemana kadar tekrar et.

Önce	Sonra	
[] [7 , 3 , 5 , 1 , 2]	[1] [3 , 5 , 7 , 2]	1 ve 7 değıştirildi
[1] [3 , 5 , 7 , 2]	[1 , 2] [5 , 7 , 3]	2 ve 3 değıştirildi
[1 , 2] [5 , 7 , 3]	[1 , 2 , 3] [7 , 5]	3 ve 5 değıştirildi
[1 , 2 , 3] [7 , 5]	[1 , 2 , 3 , 5] [7]	5 ve 7 değıştirildi
[1 , 2 , 3 , 5] [7]	[1 , 2 , 3 , 5 , 7] []	son



Seim Sıralaması (Selection Sort)

```
25 void selectionSort(int dizi[],int n)
26 {
27     int i,j;
28     int index, enkucuk;
29     for (i = 0; i < n - 1; i++)
30     {
31         enkucuk = dizi[n - 1];
32         index = n - 1;
33         for (j = i; j < n - 1; j++)
34         {
35             if (dizi[j] < enkucuk)
36             {
37                 enkucuk = dizi[j];
38                 index = j;
39             }
40         }
41         dizi[index] = dizi[i];
42         dizi[i] = enkucuk;
43     }
44 }
```

Seim Sıralaması (Selection Sort)

```
1 #include <stdio.h>
2
3 void selectionSort(int [],int);
4 int main(void)
5 {
6     int i=0,a[5];
7     printf("Sıralamak istediğın 5 sayı gir\n");
8     while(i<5){
9         scanf("%d",&a[i]);
10        i++;
11    }
12    i=0;
13    selectionSort(a,5);
14
15    printf("Selection sort işleminden sonra...\n");
16    while(i<5){
17        printf("%d ",a[i]);
18        i++;
19    }
20    return 0;
21 }
```

Arama

- ▶ Bir dizi içerisinde belli bir elemanı bulma sürecine arama denir.
- ▶ İki arama tekniği vardır:
 - Doğrusal Arama (Linear Search)
 - İkili Arama (Binary Search)



Doğrusal Arama (Linear Search)

- ▶ Dizinin her bir elemanını aranan ile karşılaştırır.
- ▶ Dizi sıralı değilse, aranan ilk elemanda olabilir, son eleman da.
- ▶ En kötü durumda N elemanlı dizide algoritmanın karmaşıklığı $O(n)$ dir.
- ▶ Büyük boyutlu dizilerde kullanımı uygun değildir.



Doğrusal Arama (Linear Search)

```
21 int linearSearch(int dizi[],int aranan,int n)
22 {
23     for (int i = 0; i < n; i++)
24     {
25         if (dizi[i] == aranan)
26             return i;
27     }
28     return -1;
29 }
```

Doğrusal Arama (Linear Search)

```
1 #include <stdio.h>
2
3 int linearSearch(int [],int,int);
4 int main(void)
5 {
6     int dizi[] = { 1, 3, 5, 7, 8, 10, 11 };
7     int sonuc, aranan,i;
8     for (i = 0; i < 7; i++)
9         printf("%d ",dizi[i]);
10
11     printf("Arananı giriniz:");
12     scanf("%d",&aranan);
13
14     sonuc = linearSearch(dizi, aranan,7);
15     if (sonuc == -1)
16         printf("\nAranan dizide yok\n");
17     else
18         printf(sonuc + ". sırada bulundu\n");
19 }
```

• Gelecek Hafta

- ▶ Struct, Enum and Typedef
- ▶ Bir Bağlı Doğrusal Listeler



Kaynaklar

- ▶ Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- ▶ Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- ▶ Paul J. Deitel, “C How to Program”, Harvey Deitel.
- ▶ “A book on C”, All Kelley, İra Pohl



S o r u l a r
?



Dinlediğiniz için teşekkürler

CANER ÖZCAN



canerozcan@karabuk.edu.tr