CME 112- Programming Languages II

Week 7
String Functions
Sorting & Searching

Assist. Prof. Dr. Caner Özcan

Science is trying to understand the language of nature. Those who understand the language are friendly to nature, and those who do not understand are enemies

String

A string is an array of characters terminated by the NULL char ‘\0’

Example: char str[8];

- Declares a char array that contains at most 8 chars
- If char array str will be used to store strings, it can contain at most 7 chars and MUST be terminated by the NULL char ‘\0’

String İşlemleri

- C standard library provides many functions to manipulate strings.
- You need to include <string.h> to use these functions

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

- Here are some of the important functions:
  - **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

Sorting

- Placing a group of data in descending or ascending order.
- Sorting data is very useful for computer systems.
- Makes searching and listing a group of data faster and easier.
- Most popular sorting algorithms:
  - Insertion sort
  - Selection sort
  - Bubble sort
  - Quick sort

Bubble Sort

- Time complexity is $O(n^2)$
- If c number of item of n items is not sorted time complexity is O(c n)
- Design of algorithm is easy, but algorithm is not efficient.
- Can be used for small size lists or lists having mostly sorted items.

Bubble Sort

Array to be sorted : [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]**

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  }
```

(n-1);

Bubble Sort


```c
#include <stdio.h>

void bubbleSort(int [],int);
int main(void)
{
    int i=0,a[5];
    printf("Siralamak istediğin 5 sayi gir\n");
    while(i<5){
        scanf("%d",&a[i]);
        i++;
    }
    i=0;
    bubbleSort(a,5);

    printf("Bubble sort isleminden sonra...\n");
    while(i<5){
        printf("%d ",a[i]);
        i++;
    }
    return 0;
}
```

Insertion Sort

- Appropriate for inserting an item into an already sorted list of data.
- Complexity of inserting an item into an already sorted list of data: O (n)
- If list or array is not sorted complexity: O (n^2)

Insertion Sort

Array to be sorted : [7,3,5,8,2]

Initial state : [7][3,5,8,2]

Before	After
[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][ ]

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  }
```

Insertion Sort


```c
#include <stdio.h>

void insertionSort(int [],int);
int main(void)
{
    int i=0,a[5];
    printf("Siralamak istediğin 5 sayi gir\n");
    while(i<5){
        scanf("%d",&a[i]);
        i++;
    }
    i=0;
    insertionSort(a,5);

    printf("Insertion sort isleminden sonra...\n");
    while(i<5){
        printf("%d ",a[i]);
        i++;
    }
    return 0;
}
```

Selection Sort

- If an item is in its true place it does not change its order.
- Change of items is less in half sorted group of data.
- Take the first item in the list and exchange with the minimum item of others.

  Repeat this until the last item.

Before	After	
`[][7,3,5,1,2]`	`[1][3,5,7,2]`	`1 and 7 exchanged`
`[1][3,5,7,2]`	`[1,2][5,7,3]`	`2 and 3 exchanged`
`[1,2][5,7,3]`	`[1,2,3][7,5]`	`3 and 5 exchanged`
`[1,2,3][7,5]`	`[1,2,3,5][7]`	`5 and 7 exchanged`
`[1,2,3,5][7]`	`[1,2,3,5,7][]`	`end`

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  }
```

Selection Sort


```c
#include <stdio.h>

void selectionSort(int [],int);
int main(void)
{
    int i=0,a[5];
    printf("Siralamak istediğin 5 sayi gir\n");
    while(i<5){
        scanf("%d",&a[i]);
        i++;
    }
    i=0;
    selectionSort(a,5);

    printf("Selection sort isleminden sonra...\n");
    while(i<5){
        printf("%d ",a[i]);
        i++;
    }
    return 0;
}
```

Search

- The process of finding a particular element of an array is called searching.
- Two searching techniques will be discussed
  - Linear Search
  - Binary Search

Linear Search

- Compares each element of the array with the search key.
- Since the array is not in any particular order, it is just as likely that the value will be found in the first element as in the last.
- In the worst case with N number of elements, the algorithm's complexity is O(N)
- It should not be used in large size arrays.

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  }
```

Linear Search

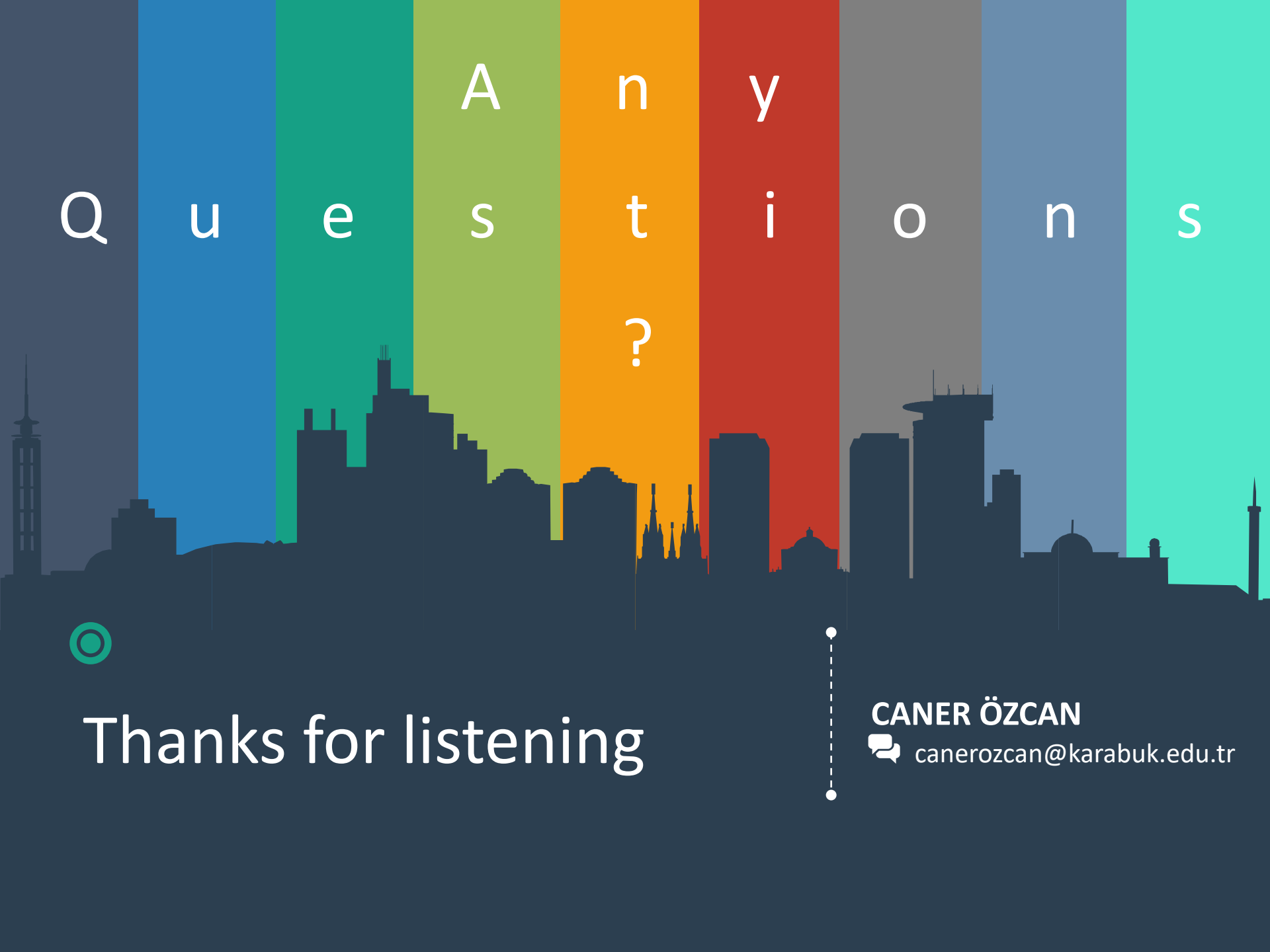

```c
#include <stdio.h>

int linearSearch(int [],int,int);
int main(void)
{
    int dizi[] = { 1, 3, 5, 7, 8, 10, 11 };
    int sonuc, aranan,i;
    for (i = 0; i < 7; i++)
        printf("%d ",dizi[i]);

    printf("Arananı giriniz:");
    scanf("%d",&aranan);

    sonuc = linearSearch(dizi, aranan,7);
    if (sonuc == -1)
        printf("\nAranan dizide yok\n");
    else
        printf(sonuc + ". sırada bulundu\n");
}
```

Next Week

- Struct, Enum and Typedef
- Singly Linked Linear Lists

References

- 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



Thanks for listening

CANER ÖZCAN
canerozcan@karabuk.edu.tr

Dip not..

Aklınızla, bedeninizle masanın başında oturup okumanızı tavsiye ederim.
Okumanız, düşünmeniz ve
düşünüp fikirlerinizi geliştirmeniz
gerekmektedir.

Prof. Dr. Fuat Sezgin