Uygulama Notları: 12

FİZ219 - Bilgisayar Programlama I | 25/01/2020

String Değişkenleri

- Giriş
- Tanımlama
- Kıyaslama
- Yeni string değişkenleri oluşturmak
  - Dizi Muamelesi çekmek
  - strcat() ve cstrcat() fonksiyonları
  - printf() fonksiyonu ve string değişkenleri
  - sprintf() fonksiyonu
- Meydan Okuma (Challenge)

Emre S. Tasci emre.tasci@hacettepe.edu.tr (mailto:emre.tasci@hacettepe.edu.tr)

Giriş

Şu ana kadar Octave ile ağırlıklı (neredeyse istisnasız) olarak sayılar üzerine çalıştık ki, Octave'ın mühendislik uygulamalarında kullanmak üzere geliştirildiğini düşünürsek bu zaten beklenen bir şeydi. Fakat bazen, özellikle de başka programlarda kullanacağımız çıktılar oluşturmak istediğimiz zaman, çıktımızı belirli bir biçimde sunmak isteriz. `printf()` fonksiyonunu bu işler kullanabileceğimizi gördük ama şimdiye kadar ele aldığımız tüm kullanımlarda yazı kısmı sabit olup, sayısal değerler değişiyordu, örneğin:

In [1]:

```
for i = 3:8
    printf("Elimizdeki sayi: %d --> Bu sayi ",i)
    if(mod(i,2)==0)
        printf("cift bir sayidir.\n");
    else
        printf("tek bir sayidir.\n");
    endif
endfor
```

```
Elimizdeki sayi: 3 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 4 --> Bu sayi cift bir sayidir.
Elimizdeki sayi: 5 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 6 --> Bu sayi cift bir sayidir.
Elimizdeki sayi: 7 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 8 --> Bu sayi cift bir sayidir.
```

Octave'da -çok gelişmiş olmasa da- sözcükler ve harfler için de bir değişken türü vardır. Harf dizilerinden (sözcük) ve sözcük takımlarından (cümle) oluşan bu değişkenler **string** değişkeni olarak adlandırılırlar. Tanım itibarı ile dizidirler.

Tanımlama

String değişkenleri tırnak (") içinde tanımlanır.

In [2]:

```
isim = "Batuhan"
```

```
isim = Batuhan
```

şeklinde değeri "Batuhan" olan, `isim` adında bir string değişkeni tanımladığımızda, bu aslıdan 7 elemanı olan bir dizidir. Bir sayı dizisinin çeşitli elemanlarına nasıl ulaşabiliyorsak, string değişkeninin (*harf dizisinin*) elemanlarına da aynı şekilde ulaşabiliriz:

In [3]:

```
isim(2) % 2. eleman (harf)
isim(4:6) % 4.,5.,6. harfler
isim(5:end-1) % 5. harften sondan bir önceki harfe kadar
isim(1:2:7) % 1. harften 7. harfe bir atlayarak (2 ekleyerek)
isim(3:-1:1) % 3. harften 1. harfe geriye doğru
```

```
ans = a
ans = uha
ans = ha
ans = Bthn
ans = taB
```

Her harfin bir sayısal değeri vardır, bu değerler için ASCII standardı verilen bir değerler tablosu kullanılır. Örneğin "A" karakterine 65; "B"ye 66; "a"ya 97; "b"ye 98 karşılık gelir. Bu karşılıkları printf'e bir string değişkeni besleyip, çıktıyı tam sayı olarak isteyerek görebiliriz:

In [4]:

```
str1 = "ABCabc";
printf("%d\n",str1);
```

```
65
66
67
97
98
99
```

ASCII tablosu 128 (2^7) değerden oluşup, 60'lı yıllarda bilgisayar sistemleri arasında uyum olması için standartlaştırılmıştır. Sadece yazılı karakterleri değil, bilgisayara verilecek işaretleri (*"satır sonu", "sekme", "escape (çıkış) sinyali", vs.*) de içerir:

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Kaynak: Wikipedia (https://en.wikipedia.org/wiki/File:ASCII-Table.svg)

Sonrasında (günümüzde) UTF-8 standardı kabul edilip, öntanımlı karakter sayısı 1112064'e çıkarılmıştır (UTF-8'in ilk 128 karakteri ASCII tablosuna karşılık gelmektedir -- aslında Octave'da UTF-8 desteği vardır -yani Türkçe karakterleri de destekler- ama işleri karıştırmamak adına 128'lik ASCII'de duracağız ;).

Sayısal değerleri `char` komutu ile harflere dönüştürebiliriz. Örneğin, yukarıdaki tablodan bakarak "Fiz219" yazalım (rakamların da tıpkı harfler gibi farklı indislere sahip olduğuna dikkat edin -- mesela 1 -> 49):

In [5]:

```
dizi = [70, 105, 122, 50, 49, 57];
str2 = char(dizi)
```

```
str2 = Fiz219
```

Bu sayı <-> harf karşılığından ötürü, harf dizimizi doğrudan dizi olarak da tanımlayabilirdik:

In [6]:

```
str3 = ["F", "i", "z", "2", "1", "9"]
```

```
str3 = Fiz219
```

Neyse ki, her seferinde bu kadar uğraşmamak için, başta gösterdiğimiz tırnak içinde tanım yöntemi var:

In [7]:

```
str4 = "Fiz219"
```

```
str4 = Fiz219
```

Aynı şekilde, tırnak(") yerine kesme (') işareti de kullanılabilmektedir.

Kıyaslama

İki sayıyı büyüklük, küçüklük ve eşitlik kriterleri ile nasıl kıyaslayabileceğimizi biliyoruz: <,>,==,!=,<= ve >= operatörleri ile. Bu operatörleri harf bazında string değişkenlerine de uygulayabiliriz:

In [8]:

```
% "a", "b"den büyük mü (yani sonra mı geliyor)?
"a" > "b"
```

```
ans = 0
```

In [9]:

```
% "a", "b"den küçük mü (yani önce mi geliyor)?
"a" < "b"
```

```
ans = 1
```

In [10]:

```
% "a", "b"ye eşit mi (yani ASCII tablosunda aynı yerde mi)?
"a" == "b"
```

```
ans = 0
```

İşler buraya kadar iyi idi, hatta bu değerleri değişkenlere atadığımızda da yolunda gidiyor gibi görünüyor:

In [11]:

```
str5 = "a"
str6 = "b"
str5 > str6
str5 < str6
str5 == str6
```

```
str5 = a
str6 = b
ans = 0
ans = 1
ans = 0
```

Fakat, string değişkenlerimiz birden fazla harften oluştuğu zaman işler biraz karışıyor:

In [12]:

```
str7 = "Bilgisayar "
str8 = "Programlama"
str7 > str8
```

```
str7 = Bilgisayar
str8 = Programlama
ans =

  0  0  0  0  0  1  0  1  0  1  0
```

Aslında karışan bir şey yok, Octave, kendinden beklendiği üzere, verilen iki diziyi (string değişkenlerinin harf dizileri olduğunu hatırlayın), eleman bazında kıyaslıyor. Burada aslında şu soruyu sormuş olduk: "Bilgisayar " kelimesinin (*harf **dizisinin***) hangi harfleri alfabede (*ASCII Tablosunda*) "Programlama" kelimesinde karşılık gelen harften sonra yer almaktadır? Cevap olarak bize verdiği: "1" (*Doğru*) olarak döndürülen, 6.,8. ve 10. harfler, yani: "s","y" ve "r" harfleri:

In [13]:

```
str7(str7>str8)
```

```
ans = syr
```

Octave açısından "Bilgisayar " ve "programlama" kelimelerini kıyaslamakla, [66, 105, 108, 103, 105, 115, 97, 121, 97, 114, 32] ile [112, 114, 111, 103, 114, 97, 109, 108, 97, 109, 97] dizilerini kıyaslamak arasında hiçbir fark yok.

Bunu görünce, iki string değişkeninin birbirine eşitliğinin sorgulanmasının da pek istediğimiz gibi sonuçlanmayacağı ortada. Nitekim:

In [14]:

```
str7 == str8
```

```
ans =

  0  0  0  1  0  0  0  0  1  0  0
```

evet, gerçekten de iki kelimenin 4. ("g") ve sonran iki önceki "a" harfleri birbirine eşit ama bizim aklımızdaki iki kelimenin tümüyle birbirine eşit olup olmadığı idi. Tabii ki bunu, verilen bir dizinin *tüm* elemanlarının sıfırdan farklı olup olmadıklarını döndüren `any()` fonksiyonunu kullanarak 2 adımda yapabiliriz ama her seferinde bu kadar uğraşmayalım diye, `strcmp()` (Sözcük kıyasla: "{str}ing {c}o{mp}are") diye bir fonksiyon üretmişler:

In [15]:

```
% str7 değişkeninde tutulan kelime, str8'dekiyle aynı mı?
strcmp(str7,str8)
```

```
ans = 0
```

Hazır strcmp'u kullanmaya başlamışken, yukarıda farklı farklı şekillerde tanımladığımız string değişkenlerinin birbirlerine eşitliğini de kontrol edelim:

In [16]:

```
dizi = [70, 105, 122, 50, 49, 57];
str2 = char(dizi)

str3 = ["F", "i", "z", "2", "1", "9"]
str4 = "Fiz219"
```

```
str2 = Fiz219
str3 = Fiz219
str4 = Fiz219
```

In [17]:

```
strcmp(str2, str3)
strcmp(str3, str4)
```

```
ans = 1
ans = 1
```

Görüldüğü üzere, tanım yöntemleri birbirine eşdeğer. Bir de büyük/küçük harf olayı var. Biliyoruz ki, büyük harflerle küçük harfler ASCII tablosunda farklı yerlere karşılık geldiğinden (örn: "A" : 65 & "a" : 97) eşitlik olmayacak ama bu da çoğu zaman çok da arzu ettiğimiz bir şey değil. Bu işin de üstesinden -en azından İngiliz alfabesindeki harflerde- büyük/küçük harf ayrımı yapmayan `strcmpi()` fonksiyonu geliyor (sondaki "i", *Büyük/küçük durumu yoksay*'ın İngilizcesi *{i}gnore case*'den geliyor):

In [18]:

```
str4 = "Fiz219"
str9 = "FIZ219"
str10 = "FIz219"

strcmp(str4,str9)
strcmp(str4,str10)

strcmpi(str4,str9)
strcmpi(str4,str10)
```

```
str4 = Fiz219
str9 = FIZ219
str10 = FIz219
ans = 0
ans = 0
ans = 1
ans = 1
```

Yeni string değişkenleri oluşturmak

Dizi muamelesi çekmek

Yeni string değişkenleri oluşturmak için, öncelikle elimizdeki mevcut string değişkenlerinden yola çıkabiliriz. Örneğin:

In [19]:

```
str1 = "Merhaba"
str2 = "Dünya!"
```

```
str1 = Merhaba
str2 = Dünya!
```

şeklinde iki değişkenimiz olsun, biz de "Merhaba Dünya!" şeklinde bunların birleşimi olan üçüncü bir değişken kurmak istiyoruz. Birinci yöntem, klasik, string'leri harf dizisi olarak ele almak, nasıl ki iki diziyi köşeli parantezler içerisinde toplayıp birleştirebiliyorduk, burada da çalışması lazım:

In [20]:

```
sayi1 = [1 2 3]
sayi2 = [5 6 7]
sayi3 = [sayi1 sayi2]
```

```
sayi1 =

   1   2   3

sayi2 =

   5   6   7

sayi3 =

   1   2   3   5   6   7
```

Harf dizilerimize de uygulayalım bakalım:

In [21]:

```
str1 = "Merhaba"
str2 = "Dünya!"
str3 = [str1 str2]
```

```
str1 = Merhaba
str2 = Dünya!
str3 = MerhabaDünya!
```

Kelimelerin birbirine yapışık olması dışında fena değil! Araya koyacağımız boşluğu da bir eleman olarak ele alıp bir daha deneyelim:

In [22]:

```
str3 = [str1 " " str2]
```

```
str3 = Merhaba Dünya!
```

strcat() ve cstrcat() fonksiyonları

Yine çok uğraştırmamak adına, Octave'da bu birleştirme işlerini yapan öntanımlı bir fonksiyon da mevcuttur: `strcat()` (*string birleştir* : *{str}ing con{cat}anate*)

In [23]:

```
str4 = strcat(str1,str2)
```

```
str4 = MerhabaDünya!
```

In [24]:

```
% sondaki bosluklar, sekmeler, vs. strcat tarafindan yoksayilir:
str5 = strcat("Merhaba ","Dunya")
```

```
str5 = MerhabaDunya
```

In [25]:

```
% bastaki bosluklar, sekmeler, vs. strcat tarafindan ele alinir:
str6 = strcat("Merhaba"," Dunya")
```

```
str6 = Merhaba Dunya
```

Eğer sondaki boşlukları korumak istiyorsanız, `cstrcat()` komutunu kullanabilirsiniz:

In [26]:

```
strcat("Merhaba ","Dunya")
cstrcat("Merhaba ","Dunya")
disp("-----------------")
strcat("Merhaba"," ","Dunya")
cstrcat("Merhaba"," ","Dunya")
```

```
ans = MerhabaDunya
ans = Merhaba Dunya
-----------------
ans = MerhabaDunya
ans = Merhaba Dunya
```

printf() fonksiyonu ve string değişkenleri

Çıktı biçimlemede sıkça kullandığımız `printf()` fonksiyonumuz, şaşırtıcı olmayan bir biçimde, sayılara ilave olarak string değişkenlerini de destekler. Yapmamız gereken, değişkenimizin string olacağını belirtmek için '%s' yertutucusunu kullanmaktır:

In [27]:

```
printf("Dersimizin adi: %s.\n","FIZ219 - Bilgisayar Programlama");
```

```
Dersimizin adi: FIZ219 - Bilgisayar Programlama.
```

Çok da faydalı bir şey olacakmış gibi görünmüyor ama bir de şu kullanımına bakalım: baştaki tek/çift ayırıcısını hatırlayın, onu bir daha yazalım:

In [28]:

```
for i = 3:8
    if(mod(i,2)==0)
        tekcift = "cift";
    else
        tekcift = "tek";
    endif
    printf("Elimizdeki sayi: %d --> Bu sayi %s bir sayidir.\n",i,tekcift)
endfor
```

```
Elimizdeki sayi: 3 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 4 --> Bu sayi cift bir sayidir.
Elimizdeki sayi: 5 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 6 --> Bu sayi cift bir sayidir.
Elimizdeki sayi: 7 --> Bu sayi tek bir sayidir.
Elimizdeki sayi: 8 --> Bu sayi cift bir sayidir.
```

Madem printf()'i kullanıyoruz, onun nimetlerinden de faydalanabiliriz, örneğin sabit bir yer ayırabiliriz veya sağa/sola yaslayabiliriz:

In [29]:

```
str = "Deneme";
printf("Denemeye 10 karakter ayarlayalim:%10s.\n",str) % Saga yaslanmis
printf("Denemeye 10 karakter ayarlayalim:%-10s.\n",str)% Sola yaslanmis
```

```
Denemeye 10 karakter ayarlayalim:    Deneme.
Denemeye 10 karakter ayarlayalim:Deneme    .
```

Sizce `printf` fonksiyonunun çıktısı nedir? İçinde harfler var, dolayısıyla doğru tahmin ettiniz: printf, string türünde çıktı verir!

Fakat verdiği bu çıktı, sadece ekrana yazdırmak içindir. Bu farkı anlamak için aşağıdaki fonksiyonu inceleyelim:

In [30]:

```
function f = topla(a,b)
a+b
endfunction

topla(3,5)
```

```
ans =  8
```

Görünürde gayet eli yüzü düzgün bir fonksiyon olup, verilen iki sayıyı topluyor. Peki, biz o toplamı alıp bir başka yerde kullanabiliyor muyuz? Düşünün bakalım...

Maalesef hayır, çünkü fonksiyonun döndürdüğü hiçbir değer yok! Hesabı yapmasına yapıyor ama döndürmüyor. Yani:

```
toplam = topla(3,5)
```

deyince, bir ihtimal tahmininizin aksine, `toplam` değişkeninin değerini 8 yapmıyor (ekrana 8 yazsa dahi):

In [31]:

```
toplam = topla(3,5)
```

```
ans =  8
error: value on right hand side of assignment is undefined
```

Fonksiyona tekrar baktığımızda, dönüş değerini (bu örnekte "f") tanımlamamış olduğumuzu fark ediyoruz, doğrusu şöyle olmalıydı:

In [32]:

```
function f = topla(a,b)
f = a+b
endfunction

toplam = topla(3,5)
```

```
f =  8
toplam =  8
```

`printf` 'in yaptığı da buna benzer bir şey: kendisine verilen değişkenleri güzelce biçimler, ekrana yazar ama döndürdüğü değer aslında ekrana yazdığı şey değil de, toplam karakter sayısıdır:

In [33]:

```
str1 = "Merhaba"
str2 = "Dunya!"
% Merhaba Dunya!
% 12345678901234  : Bastan sona toplam 14 karakter
donus_degeri = printf("%s %s",str1,str2)
```

```
str1 = Merhaba
str2 = Dunya!
Merhaba Dunya!donus_degeri =  14
```

sprintf() fonksiyonu

...keşke `printf` le yazdığımız string çıktılarını ekrana değil de, bir string değişkenine yönlendirebileceğimiz bir yol olsaydı... Aaaa, böyle bir yol var aslında! **`sprintf()`** fonksiyonu!!!

In [34]:

```
str1 = "Merhaba"
str2 = "Dunya!"
donus_degeri = sprintf("%s %s",str1,str2)
donus_degeri
```

```
str1 = Merhaba
str2 = Dunya!
donus_degeri = Merhaba Dunya!
donus_degeri = Merhaba Dunya!
```

Bu şekilde artık istediğimiz -hemen hemen- her şeyi yapabiliriz, mesela:

In [35]:

```
printf("%s %s %s\n",donus_degeri,donus_degeri,donus_degeri)
```

```
Merhaba Dunya! Merhaba Dunya! Merhaba Dunya!
```

(Bu yukarıdaki örnek çok da aydınlatıcı olmadı, şunu deneyelim bir de:)

In [36]:

```
function f = merhaba(isim)
f = sprintf("Merhaba %s!\n",isim);
endfunction

md = merhaba("Dunya")
me = merhaba("Emre")
```

```
md = Merhaba Dunya!

me = Merhaba Emre!
```

Meydan Okuma (*Challenge*)

Daha gelişmişini size "meydan okuma" olarak vereyim:

Verilen 4 basamaklı bir sayıyı yazı olarak döndüren bir fonksiyon yazın.

Örneğin:
`sayi2yazi(1234)` -> *"bin iki yüz otuz dört"*
döndürsün.