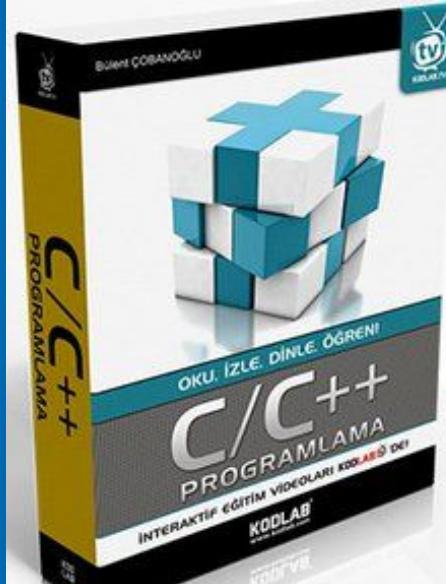


C / C++ Eğitimi



Bülent COBANOĞLU

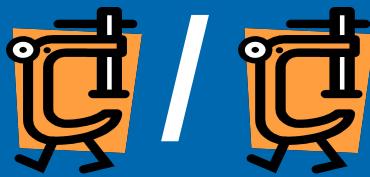
C/C++ PROGRAMLAMA



Akademi: **DEPAR AKADEMİ**

<https://www.deparakademi.com.tr/>

Eğitmen: Bülent Cobanoğlu (Bülend Hoca)



C / C++ Eğitimi-3

- --: Diziler (Arrays)
- --: Matrisler (İki Boyutlu Diziler)
- --: Karakter Dizileri
- -: String Dizileri
- -: struct veri yapısı
- --: C++ String işlemler ile ilgili fonksiyonlar
- --: Fonksiyonlar (Functions)
- --: Fonksiyonlara Dizi Aktarımı
- --: Özyinelemeli Fonksiyonlar (Recursion)
- --: Rastgele Sayı Üretimi
- -: Matematiksel Fonksiyonlar
- -: #define ile fonksiyon tanımlama

Bülend Hoca (Dr. Bülent Çobanoğlu)

DİZİLER (Arrays)

- Bellekte sürekli yer kaplayan art arda sıralanmış aynı türden verilerin oluşturduğu kümeye **dizi (array)** denir
 - Diziler bir grup ardışık bellek bölgesidir.
 - Dizideki her eleman aynı veri tipindedir
- Tanımlamada önce **veri tipi**, **dizi_adı** ve **[]** parantezi arasına toplam eleman adedi belirtilir.
 - `int dizi[10];`
- Eleman erişimi **[]** parantezi arasındaki tam sayı ile belirtilir ve bu sayıya **sıra numarası** (indis, ingilizce: **index**) denir.
 - `dizi[5]=3;` // diziye değer atama (yazma)
 - `x=dizi[3];` // dizi değerini okuma
- Dizi, **0** sıra (indis) numarası ile başlar ve pozitif sayılardan oluşur.

Dizi Tanımlamaları

Tamsayı dizi örnekleri (<i>İçerisinde tamsayıları sakladığımız 6 elemanlı bir A dizisi örnekleri</i>)	<ul style="list-style-type: none">int A [6] ;int n=6; int A[n] ;int A[]={8, 9, 12, 4, 1, 0} ;
Karakter dizisi örnekleri	<ul style="list-style-type: none">char A[] = "ali";char A[] = { 'a', 'l', 'i', '\0' } ;
String dizi örnekleri	<ul style="list-style-type: none">char Ad[eleman Sayısı][Max. Karakter uzunluğu];char Ad[3][5]={"Paz", "Sal", "Ali"} ;
String dizi örnekleri	<ul style="list-style-type: none">char *gun [7] ;char gun[] [7]= {"Paz", "Salı", "Çarş", "Per"} ;
C++ özgү string dizi tanımlaması	string renk [] = { "Mavi", "Sarı", "Mor", "Siyah" } ; //string.h kütüphanesini ister

■ String ifade = Karakter dizisi

Dizi Tanımlamaları-2

Başlangıç Değerleri Belli Olan Tanımlamalar

```
int x[ ]={10,20,30,40,50};
```

x[0] □ 10
x[1] □ 20
x[2] □ 30
x[3] □ 40
x[4] □ 50

```
float x[5]={0,1.4,1.5,1.6};
```

x[0] □ 0.0
x[1] □ 1.4
x[2] □ 1.5
x[3] □ 1.6
x[4] □ 0.0

```
char kelime[5]={'C','+', '+'};  
kelime[0] □ C  
kelime[1] □ +  
kelime[2] □ +  
kelime[3] □ \0 veya null  
kelime[4] □ \0 veya null
```



```
int x[5]={0};  
x[0] □ 0  
x[1] □ 0  
x[2] □ 0  
x[3] □ 0  
x[4] □ 0
```

```
char a[5]="C\C++"; // veya char *a="C\C++";  
Bu tanımda sorun var mı?
```

NULL Karakter

- Karakter dizileri için sonlandırıcı karakter ASCII tablosunun sıfır numaralı ('\0') karakteridir.
- C dilinde bu karakter NULL sembolik sabiti ile eşleştirildiği için yerine NULL deyimi de kullanılabilir.

Karakter Dizileri en son elemanı

- Bir karakter dizisinin en son elemanı **NULL** ('**\0**') karakteridir.
 - `char s[10] = { 'a', 'l', 'i', '\0' };`
- `char` türden bir dizide null karakter görene kadar ilerlemek için şu kalıplar kullanılabilir:

```
for (i = 0; s[i] != '\0'; ++i) {...}
```

//veya

```
while (s[i] != '\0') {...}
```

SORU 40



```
int main() {
    int a[10] = {2, 5, 1, 9, 4, 8, 6, 10};
    printf("%d\n", a[a[1]]);
}
```

Yukarıda verilen program hangi çıktıyi üretir?

- A)** 2
- B)** 5
- C)** 8
- D)** 0
- E)** Çalışma zamanı hatası oluşur.

Dizinin Bellekte Kapladığı Alanın Tespiti: **sizeof** Operatörü

- Değişken, dizi ve yapıların bellekte kapladıkları toplam alanı belirlemek için **sizeof** operatörü kullanılır.
- Kullanım şekli; **sizeof(diziAdı); sizeof(tip);**
- Örneğin aşağıdaki kod satırlarının;
- int dizi[5] = {3, 4, 5, 6, 7};**
- printf("Dizinin kapladığı alan.:%d", sizeof(dizi));**

//bellekteki kapladığı alan kaç byte dır?

İpucu: sizeof(int) == 4 ise sizeof(dizi) = ?

Dizinin Bellekte Kapladığı Alanın Tespiti: **sizeof** Operatörü

- ❑ Aşağıdaki program parçası ekrana ne yazar?

```
char dizi[]="Italya";
printf("%d", sizeof(dizi)/sizeof(dizi[0]));
```

struct veri yapısı

- Elemanları bellekte ardışık biçimde tutulan fakat farklı veri tiplerinin belirli bir mantık içerisinde anlamlı bir topluluk oluşturduğu veri yapılarına “struct(yapı)” adı verilir.
- Temel veri tiplerinin bir isim altında birleştirilmesi ile kullanıcı tanımlı özel yapılar oluşturabilirsiniz.
- Dizi elemanları aynı türden olduğu halde yapı elemanları farklı veri tiplerinden oluşabilmektedir.

```
struct DATE {  
    int day, month, year;  
};
```

Yapı bildirimi yapıldıktan sonra artık o yapı türünden nesneler tanımlanır;
`struct DATE my_date;`

```
struct Rehber{  
    char Ad[15];  
    char Soyad[20];  
    char Adres [50];  
    double TelNo;  
} Kayit;  
/*'Rehber' yapısı program içinde 'Kayıt'  
değişkeni ile temsil edilir.*/
```

struct veri yapısı

- Bir dizi, struct'ın elemanı olabilir.

```
#include <stdio.h>
struct PERSONEL {
    char name[30];
    int no;
};
```

```
int main() {
    struct PERSONEL p = { "Bülend Hoca", 111 };
    printf("%s, %d\n", p.name, p.no);
    char ch = p.name[3]; //3.indisli karakter
    putchar(ch);
    printf("\n");
}
return 0;
```

p ----> struct PERSONOL türündendir
p.name -----> char * türündendir
p.name[3] -----> char türündendir

Bir yapının bellekte kapladığı alanın tespiti: **sizeof** Operatörü



- Aşağıdaki tür bildirimleri göz önüne alındığında, sizeof (int) == 4, sizeof (char) == 1 olduğunu ve bir işaretçinin (pointer) 4 bayt bellek kapladığını bilerek **sizeof (t_nodo)** hesaplayın?

```
typedef struct nodo { int valori[100];  
                      char nome[50];  
                      struct nodo *next;  
} t_nodo;
```

Matrisler

- Bellekte art arda sıralanmış satır ve sütunlardan oluşan yapıya **matris** (iki boyutlu dizi) adı verilir.
-
- Vektörlerin (tek boyutlu dizilerin) tanımlanmasında tek köşeli ‘[]’ parantez kullanılırken matrislerin tanımlanmasında ise iki köşeli ‘[[]]’ parantez kullanılır.

Çok boyutlu diziler

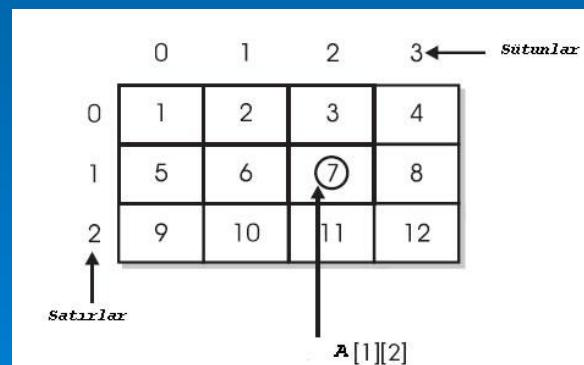


```
1 int a[10][10];           //10x10 array for 100 integers
2
3 float b[10][10][10];   //10x10x10 array for 1000 floats
```



```
1 char a[3][3] = {{'X', '0', 'X'},
2                   {'0', '0', 'X'},
3                   {'X', 'X', '0'}};
4
5 int b[2][2][2] = {{{0, 1}, {2, 3}}, {{4, 5}, {6, 7}}};
```

```
int A [4][3] = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```



Matrisler

```
int a[3][3] = {{0, 1, 2},  
                {3, 4, 5},  
                {6, 7, 8}};
```

Row, Column

a[y][x]

Row 0

a[0][0] = 0;
a[0][1] = 1;
a[0][2] = 2;
a[1][0] = 3;
a[1][1] = 4;
a[1][2] = 5;
a[2][0] = 6;
a[2][1] = 7;
a[2][2] = 8;

Row 1

Row 2

Column			
0	1	2	
0	0	1	2
1	3	4	5
2	6	7	8

Row

1

2

y

0

1

2

2 Boyutlu Diziler(Matrisler)

`int b[4][2]= {{1, 1 }, {1, 0 }, {0, 0 }, {0, 1 }};` şeklindeki
“b” matrisi için şunlar söylenebilir;

- a) 8 elemanlıdır.
- b) 4 satır 2 sütundan oluşmaktadır
- c) b matrisinin elemanları 1 ve 0 rakamlarından oluşmaktadır.

■ Aşağıdakilerden hangisi toplam 200 kayan noktalı sayı
barındıran 2 boyutlu bir diziyi tanımlar? (2015 Tübitak)

- A) real a[100,2]; B) float a[100,2];
- C) float a[100][2]; D) real a[100][2];

3 Boyutlu Diziler

```
int a[2][2][2] = {{{{0, 1}, {2, 3}},  
{{4, 5}, {6, 7}}}};
```

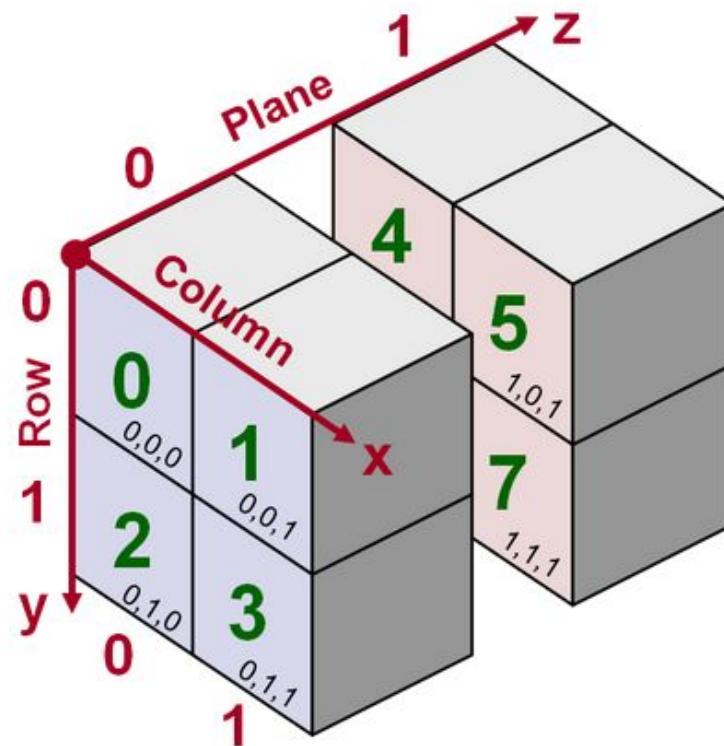
Plane, Row, Column

a[z][y][x]

Plane 0

```
a[0][0][0] = 0;  
a[0][0][1] = 1;  
a[0][1][0] = 2;  
a[0][1][1] = 3;  
a[1][0][0] = 4;  
a[1][0][1] = 5;  
a[1][1][0] = 6;  
a[1][1][1] = 7;
```

Plane 1



String Dizileri

- `char sA[] [10] = {"Ali", "Veli", "Evre"};`
- `char *sA[10] = {"Ali", "Veli", "Evre"};`

- <tip> (*<pointer_ismi>)[<sütun uzunluğu>];
- `char *sA [10];`

Burada sA ifadesi `char **` türündendir. Yani `char` türden bir işaretçiyi gösteren pointere atanabilir:

- `char **ppc;`
- `ppc = sA;`

String Dizileri

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 int main()
5 {
6     char tr[][10]={"Pazartesi","Salı", "Çarşamba", "Perşembe", "Cuma", "Cumartesi", "Pazar"};
7     char *en[10]={ "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
8     char gun[10];
9     int i=0;
10    int bayrak=0;
11    printf ("Gir Ingilizce gün ismini ..:");
12    fflush(stdout);
13    scanf ("%s",&gun);
14    for (i=0; i<7; i++){
15        //girilen gun tr dizisinde varsa
16        if (strcmpi(gun,en[i])==0) {
17            printf ("%s ...: %s",en[i], tr[i]);
18            bayrak=1;
19            break;
20        }
21    } //for sonu
22    if (bayrak==0) //tek satırlık komut için {} bloklarına gerek yoktur
23        printf ("\nsozlukte yoktur");
24    return 0;
25 }
```

C:\WINDOWS\SYSTEM32\cmd.exe

Gir Ingilizce g|n ismini ..:Friday
Friday ...: Cuma

Dizi/Matris Fonksiyon Parametreleri

```
//Parametre: Tek boyutlu dizi (vektör)
void vectorInverterRecursive ( int v[] , int n )
int confrontaStr(char s0[], char s1[])

int confrontaStr(char *s0 , char *s1) {
```

//Parametre: İki boyutlu dizi (matris)

```
void yazMatris(int(*pa)[3], int satir)
void yazMatris (int pa[][3], int satir)

void quadratoMagico(int mat [] [MAX_DIM] , int n);

void bersaglio(int m [] [N] , int n) {
```

Matrislerin Fonksiyonlara İletilmesi

```
1 #include <stdio.h>
2 //void yazMatris (int pa[][3], int satir) veya
3 void yazMatris(int(*pa)[3], int satir)
4 {
5     int i, k;
6     for (i = 0; i < satir; ++i) { //satır sayısı
7         for (k = 0; k < 3; ++k) //sütun sayısı
8             printf("%d ", pa[i][k]);
9         printf("\n");
10    }
11 }
12
13 int main()
14 {
15     int a[4][3] = { { 1, 2, 3 },
16                     { 4, 5, 6 },
17                     { 7, 8, 9 },
18                     { 10, 11, 12 } };
19     yazMatris(a, 4);
20     return 0;
21 }
22
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
1 2 3
4 5 6
7 8 9
10 11 12
```

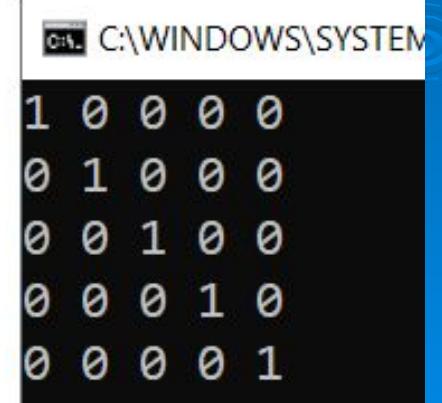
Matris Desenleri (Pattern/modello): Birim Matris

- Birim matris, köşegen (aynı satır ve sütun) elemanları 1 diğerleri 0 olan kare matristir.

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Matris Desenleri (Pattern/modello): Birim Matris

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define N 5
4 int main() {
5     int j, i;
6     int A[N][N];
7     for (i = 0; i < N; ++i) {
8         for (j = 0; j < N; ++j) {
9             if (i == j)
10                 A[i][j] = 1;
11             else
12                 A[i][j] = 0;
13             printf("%d ", A[i][j]);
14         }
15         printf("\n");
16     }
17     return 0;
18 }
```



1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

Matris Desenleri (Pattern/modello)

Sütunlar (j)

0 1 2 3 4

i	0	1	2	3	4
0	0	1	1	1	1
1	-1	0	1	1	1
2	-1	-1	0	1	1
3	-1	-1	-1	0	1
4	-1	-1	-1	-1	0

Bu A matrisini $A[i][j]$ olarak tanımlarsak;

i satır elemanlarını,

j sütun elemanlarını göstermektedir.

Birim matris örneğinin tersine $i==j$ ise matris 0, değilse 1 ya da -1 değerlerini almaktadır.

1 ya da -1 olma durumuna dikkat edersek; A matrisinin **satır elemanları, sütun elemanlarından büyük olduğunda ($i > j$ durumu) -1, diğer durumda +1 değerini** almaktadır.

Matris Desenleri (Pattern/modello)

0	1	1	1	1
-1	0	1	1	1
-1	-1	0	1	1
-1	-1	-1	0	1
-1	-1	-1	-1	0

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int j, i;
int A [5][5];
for(i=0; i < 5; ++i) {
    for(j=0; j < 5; ++j) {
        if (i==j)
            A[i][j] = 0;
        else if (i>j)
            A[i][j] = -1;
        else
            A[i][j] = 1;
        printf("%d \t",A[i][j]);
    }
    printf("\n");
}
return 0;
}
```

string işlem fonksiyonları

- ❑ C'de ismi **str** ile başlayan **strxxx** biçiminde bir grup standart fonksiyon vardır. Bu fonksiyonlara **string fonksiyonları** denir.
- ❑ Bu fonksiyonlar bir yazının başlangıç adresini parametre olarak alırlar, onunla ilgili faydalı işlemler yaparlar.
- ❑ Bu fonksiyonlar **<string.h>** kütüphanesine ihtiyaç duyar.

string işlem fonksiyonları

İşlem	C fonksiyonu;
Bir stringin karakter uzunluğunu hesaplar unsigned int strlen(char *str);	strlen(s) s.length() //C++ özgü
t Stringini null karakter görene kadar (null karakter de dahil) s ye kopyalar. char *strcpy(char *dest, char *source);	strcpy(s, t) / strncpy() //t yi s ye kopyalar. / t'nin n.karakterine kadar s'ye kopyalar copy() //C++ özgü
s ile t'yi karşılaştırır. Eğer $s < t$ ise negatif, $s == t$ ise 0, $s > t$ ise pozitif bir değer döndürür. Eğer karşılaştırma yaparken büyük – küçük harf duyarlığını istemiyorsak strcmp yerine strcmpi() / stricmp() fonksiyonları kullanılabilir.	strcmp(s, t) compare() //C++ özgü
Bir stringi ters çevirme	strrev(s1) reverse(s1.begin(), s1.end()) //C++ özgü
S stringinin ilk karakterini alma	s[0]
N elemanlı bir karakter dizisinin en son karakterini alma	s[n] s.at(n) //C++ özgü
Bir stringin sonuna yeni bir string ekler	strcat(s, t) //s'nin sonuna t'yi ekler s.append(t, 0, n) //C++ özgü
Bir string boş mu?	if (s[0] == '\0') s.empty() //C++ özgü
İki string birbirine eşit mi?/değil mi?	s1==s2 / s1!=s2

string işlem fonksiyonları

String fonksiyonları	Açıklaması
strncat(s, t, n)	s' nin sonuna n adet t ekler
strncmp (s, t, n)	s ile t'yi n. karaktere kadar karşılaştırır. strcmp() , null karakterini görene kadar bütün karakterleri karşılaştırırken, strcmp() n. karaktere kadar karşılaştırma yapar
strncpy(s, t, n)	t nin n. karakterine kadar s ye kopyalar.
strset(s, 'x')	s karakter dizisini belirtilen (bu örnek için x) karakterle doldurur.
strlwr(s)	s yi küçük harfe dönüştürür.
strupr (s)	s yi büyük harfe dönüştürür.
strchr(s, c) :	Fonksiyon s içerisinde c'yi baştan arar ve ilk bulduğu c karakterinin adresiyle geri döner, bulamazsa 0 (null) değerini döndürür.
strrchr(s, c)	s içerisinde c'yi sondan arar ve ilk bulduğu c karakterinin adresiyle geri döner, bulamazsa 0 (null) değerini döndürür.
strstr(s1, s2)	Fonksiyon s1 içerisinde s2 stringini arar. Eğer aranılan string bulunursa fonksiyonun geri dönüş değeri bulunan yerin adresidir, bulunamazsa 0 (null) değerini döndürür.
strpbrk(s1, s2)	Fonksiyon s1 içerisinde s2 stringini arar. Eğer aranılan string bulunursa fonksiyonun geri dönüş değeri ilk karakterin adresidir, bulunamazsa 0 (null) değerini döndürür.
strtok(s, "ayraçlar")	s içerisinde belirtilen ayraçlardan birine rastladığında bu karakter bir boş karakterle değiştirilir. Sonraki strtok çağrılarında arama önceki stringi sonlandıran boş karakterden sonraki karakterden başlar. Bir nevi split fonksiyonu işlevine sahiptir.
strspn(t, s)	s içerisindeki t stringinin karakter uzunluğunu verir, t stringi yoksa 0 değerini geri döndürür.

Karakter alan standart fonksiyonlar

- **gets()** : Klavyeden karakter katarlarını alır, alım Enter tuşuna basılıncaya kadar devam eder.

```
char cümle[80];
```

```
gets(cümle);
```

```
puts(cümle);
```

- **scanf()**: Klavyeden karakter katarlarını alır, alım Enter veya boşluk tuşuna basılıncaya kadar devam eder.

```
char kelime[20];
```

```
scanf("%s",kelime);
```

```
puts(kelime);
```

Not. %[^n] anlamı: Enter tuşuna basılıncaya kadar ki (tek satırlık) veriyi al

Karakter Test Fonksiyonları

C'de başı "is" ile başlayan isxxx biçiminde isimlendirilmiş bir grup standart fonksiyon vardır. Bu fonksiyonların hepsi parametre olarak bir karakter alır ve int türden bir geri dönüş değeri verir. Bu fonksiyonlar parametreleriyle aldığı karakterleri test ederler. Test doğruysa sıfır dışı bir değere yanlışsa sıfır değerine geri dönerler.

isupper (büyük harf mi?)

islower (küçük harf mi?)

isdigit ('0' ile '9' arasındaki karakterlerden biri mi?)

isxdigit (hex karakterlerden biri mi?)

isalpha (alfabetik karakterlerden biri mi?)

isalnum (alfabetik ya da nümerik (alfanümerik) karakterlerden biri mi?)

isspace (boşluk karakterlerinden biri mi? (Space, tab, new line, carriage return, vertical tab))

isascii (ASCII tablosunun ilk yarısındaki karakterlerden biri mi?)

iscntrl (İlk 32 kontrol karakterinden biri mi?)

ispunct (noktalı virgül, nokta vs. gibi karakterlerden biri mi?)

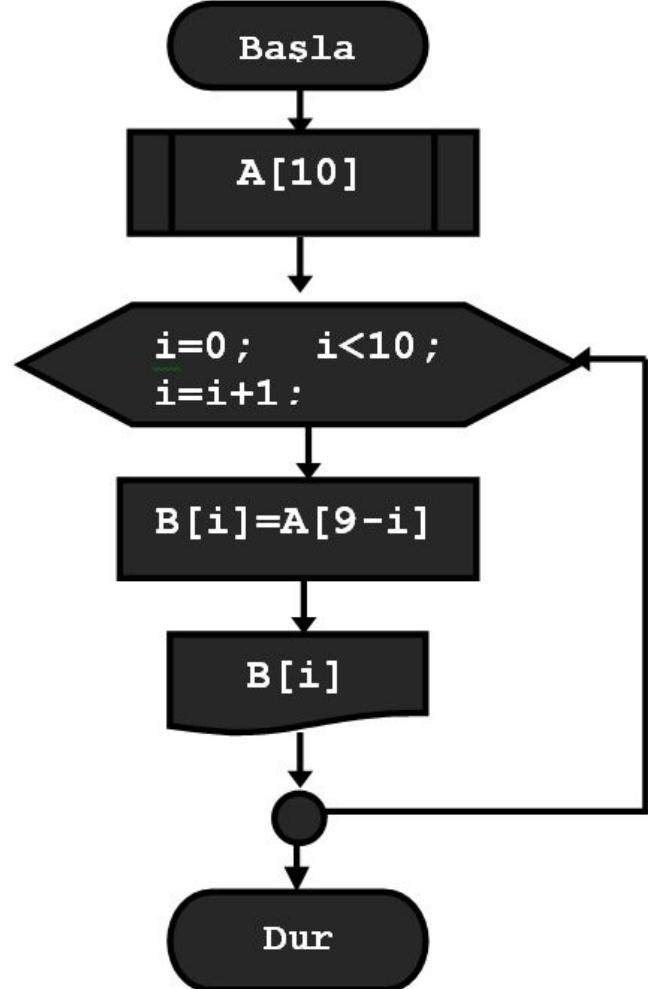
tolower (k)

k yi küçük harfe dönüştürür.

toupper (k)

k yi büyük harfe dönüştürür.

Algoritma: Bir dizi elemanlarını ters sırada başka bir diziye aktarma

Akış şeması	C Dili Kodlaması
 <pre>graph TD Basla([Başla]) --> A[A[10]] A --> Loop{ i=0 ; i<10 ; i=i+1 :} Loop --> B[B[i]=A[9-i]] B --> Bi[B[i]] Bi --> Dur([Dur]) </pre>	<pre>#include <stdio.h> #include <stdlib.h> int main() { char A[]={ 'a' , 'b' , 'c' , 'd' , 'e' , 'f' , 'g' , 'h' , 'i' , 'j' }; char B[10]; int i; printf ("A[10]="); for(i=0; i< 10; i++) { printf("\t%c", A[i]); } printf ("\nB[10]="); for(i=0; i< 10; i++) { B[i]=A[9-i]; printf("\t%c", B[i]); } return 0; }</pre>

Algoritma: Dizi elemanlarını ters yüz etme

```
1 #include <stdio.h>
2 #define SIZE    10
3 int main(void)
4 {
5     int a[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
6     int i, bos;
7     //a dizisini ters-yüz edelim
8     for (i = 0; i < SIZE / 2; ++i) {
9         bos = a[SIZE - i - 1];
10        a[SIZE - i - 1] = a[i];
11        a[i] = bos;
12    }
13
14    for (i = 0; i < SIZE; ++i)
15        printf("%d ", a[i]);
16
17 return 0;
18 }
```

C:\WINDOWS\SYSTEM32\cmd.exe

10 9 8 7 6 5 4 3 2 1



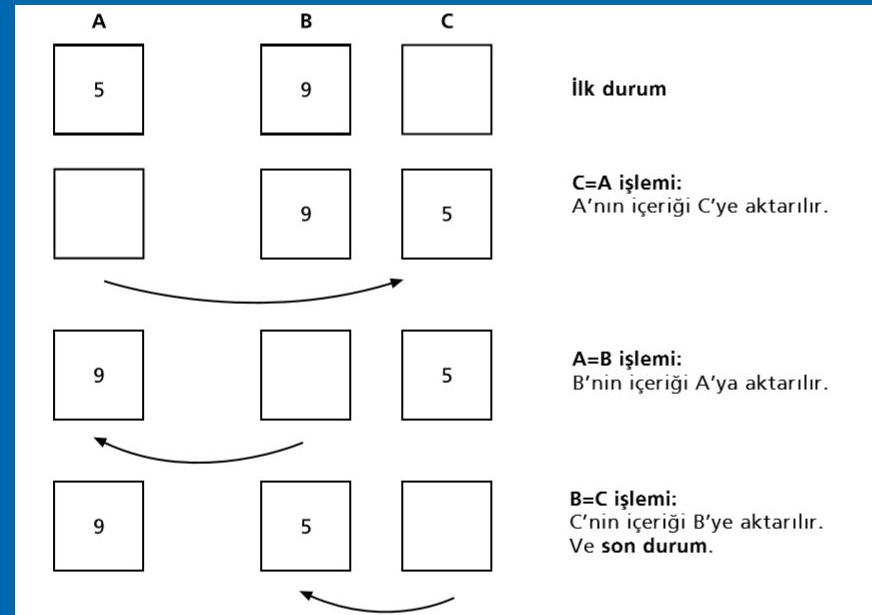
TASK: 0-100 arasındaki sayılardan Asal olanları A dizisinde tutan programı yazınız.

İstenen Çıktı:

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
59, 61, 67, 71, 73, 79, 83, 89, 97]

Bir dizinin elemanlarını yer değiştirme

İki değişkenin içeriklerini birbirine aktarma işlemi yer değiştirme olarak isimlendirilir. Bunun için üçüncü bir değişkene (geçici değişken) ihtiyacımız vardır.



Örnek; V dizisinin j. Elemanı ile bir sonraki (j+1). Elemanı aşağıdaki gibi yer değiştirilebilir;

```
aux = V[j + 1];
V[j + 1] = V[j];
V[j] = aux;
```

Fonksiyon parametresi dizi olabilir mi?

Örnek: Palindrom mu?

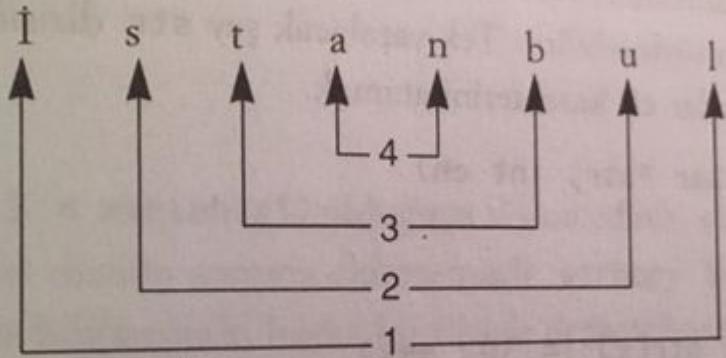
```
#include <stdio.h>
#include <string.h>
char s1[50];char s2[50];
void isPalindrom(char s1[])
{
    int x;
    strcpy(s2,s1);
    strrev(s1);
    x=strcmp(s1,s2);
    if (x==0)
        printf("\nEvet Palindrom");
    else
        printf("\nHayır değil");
}
```

```
int main() {
printf ("Bir metin gir:");
gets(s1);
printf("%s palindrom mu?", s1);
isPalindrom(s1); //fonk.çağır
return 0;
}
```

□ Bir fonksiyona
parametre olarak dizi
gönderildiğinde
adının yazılması
yeterlidir, boyutunu
belirtmeye gerek
yoktur.

Algoritma: Palindrom mu?

`strrev` fonksiyonunu nasıl tasarlayacağımız? En açık yol, baştaki elemanlarla sondakileri karşılıklı yer değiştirmektir.



Baştaki ve sondaki elemanların karşılıklı yer değiştirmeleri dizinin uzunluğunun yarısı kadar yapılmalıdır. Yani `strlen(str) / 2` kadar. Dizinin eleman sayısı tek ise ortadaki elemanın yer değiştirmeyeceği açıklıktır.

Rasgele Sayı Üretimi

- C/C++ dillerinde rastgele sayı üretimi için **rand()** fonksiyonu kullanılır. 0 ile **RAND_MAX** arasında rastgele sayı üretir.
- **RAND_MAX**, **<stdlib.h>** kütüphanesine ihtiyaç duyar.
- ANSI standartına göre **RAND_MAX** değeri 32767'dir.
- Rastgele üretilen bir sayının aralığını (0-10 gibi) belirtmek için '%' operatörü kullanılır.

Rasgele Sayı Üretimi

□ Eğer programın her çalışmasında (sürekli) farklı sayılar üretmesi isteniyorsa;

srand(time (0));

srand(time(NULL));

komut satırını yazmak gereklidir.

□ Tabi bu komut satırındaki;

time komutu; #include<time.h>

srand komutu; #include<stdlib>

kütüphanelerine ihtiyaç duymaktadır..

Rasgele Sayı Üretimi

- 0 ile N arasında 10 adet rastgele tamsayı üreten programı kodlayınız.

```
#define N 5  
for (int i = 0; i < 10; i++) {  
    x = rand() % N;  
}
```

Rasgele Sayı Üretimi

0 ile N arasında 10 adet rastgele tamsayı üreten programı kodlayınız.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #define N 50
5 int main()
6 {
7     int i, x;
8     srand(time(NULL)); //Her çalışmada farklı sayılar
9     for(i=0; i<10; i++)
10    {
11        x = (int)rand()%N;
12        printf("%d ",x);
13    }
14    return 0;
15 }
16 
```

C:\WINDOWS\SYSTEM32\cmd.exe
23 32 14 43 9 14 32 43 24 38

Rasgele Sayı Üretimi

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
#define SIZE 13

int main(){
    srand(time(0));
    int dizi[SIZE];
    //rastgele üretilen sayılar
    for(int i=0;i<13;i++){
        dizi[i] = rand()%13;
        cout<<dizi[i]<<" ";
    }

    //en sık tekrarlanan sayı
    int i, k;
    int count, max_count, max_count_val;
    for (i = 0; i < SIZE; ++i) {
        count = 1;
        for (k = i + 1; k < SIZE; ++k){
            if (dizi[i] == dizi[k])
                ++count;
            if (count > max_count) {
                max_count = count;
                max_count_val = dizi[i];
            }
        }
    }
    printf("\nMod = %d\n", max_count_val);
    return 0;
}
```

İpucu: Önce rastgele sayılar üretilir. Sonra iç içe iki döngü kullanılarak her değerin kaç tane tekrarlandığı bulunur ve bir değişkende saklanır, daha fazlası varsa yer değiştirilir. (Dizinin k'inci elemanından kaç tane olduğuna bakmak için baştan başlamaya gerek yoktur. k'inci indisten başlamak yeterlidir. Çünkü zaten k'inci indisten önce o elemandan varsa biz onun sayısını bulmuş durumda oluruz).

Rasgele Sayı Üretimi

0-10 arası rakamlardan rastgele üretilen 13 elemanlı bir dizi içerisinde en fazla yinelenen değeri bulunuz. (mod değeri).

```
1 8 2 0 11 8 2 6 1 9 9 8 3  
Mod = 8
```

İpucu: Önce rastgele sayılar üretilir. Sonra iç içe iki döngü kullanılarak her değerin kaç tane tekrarlandığı bulunur ve bir değişkende saklanır, daha fazlası varsa yer değiştirilir. (Dizinin k'inci elemanından kaç tane olduğuna bakmak için baştan başlamaya gerek yoktur. k'inci indisten başlamak yeterlidir. Çünkü zaten k'inci indisten önce o elemandan varsa biz onun sayısını bulmuş durumda oluruz).

TASK



□ Sayısal LOTO(1-90 dahil sayılardan oluşan 6 adet sayı)?



Matematiksel Fonksiyonlar

C/C++ dillerinde matematiksel fonksiyonları kullanmak için `<math.h>` kütüphanesini program başında `#include` etmek gereklidir. Bazı C++ derleyicileri (örneğin Visual C++ gibi) `M_PI` gibi bazı fonksiyonları çalıştırma için `<math.h>` kütüphanesine ek olarak program başında

```
#define _USE_MATH_DEFINES
```

komut satırını görmek ister.

Matematiksel Fonksiyonlar

Fonksiyon	Açıklama	C/C++	Çıktı
<code>fabs(x)</code>	x sayısının mutlak değerini verir.	<code>fabs(-23)</code>	23
<code>fmod(x, y)</code>	x Mod y işlemini gerçekleştirir. Mod alır.	<code>fmod(4,2)</code>	0
<code>ceil(x)</code>	x sayısını bir üst tam sayıya dönüştürür.	<code>ceil(2.5)</code>	3.0
<code>floor(x)</code>	x sayısını bir alt tam sayıya dönüştürür.	<code>floor(2.5)</code>	2.0
<code>round(x)</code>	x sayısını en yakın tam sayıya tamamlar.	<code>round(2.5)</code>	3.0
<code>rint(x)</code>	x sayısını en yakın tam sayıya tamamlar.	<code>rint(9.6)</code>	10.0
<code>cos(x)</code>	x'in radyan cinsinden trigonometrik kosinüsünü hesaplar.	<code>cos(0.0)</code>	1.0
<code>sin(x)</code>	x'in radyan cinsinden trigonometrik sinüsünü hesaplar.	<code>sin(0.0)</code>	0.0
<code>exp(x)</code>	Doğal logaritma parametresi olan e'nin üstel değerini (ex) hesaplar.	<code>exp(1.0)</code>	2.71828
<code>log(x)</code>	e tabanına göre x'in logaritmasını alır.	<code>log(2.7182)</code>	1.0
<code>log10(x)</code>	10 tabanına göre x'in logaritmasını alır.	<code>log10(20)</code>	1.30
<code>max(x, y)</code>	x ve y'nin büyük olanını verir.	<code>max(2,4)</code>	4
<code>min(x, y)</code>	x ve y'nin küçük olanını verir.	<code>min(2,4)</code>	2
<code>pow(x, y)</code>	x'in y. kuvvetini verir.	<code>pow(2,3)</code>	8
<code>sqrt(x)</code>	x'in kara kökünü verir.	<code>sqrt(9.0)</code>	3
<code>M_PI</code>	π sayısını verir.	<code>M_PI</code>	3.14159
<code>M_E</code>	e sayısını verir.	<code>M_E</code>	2.71828

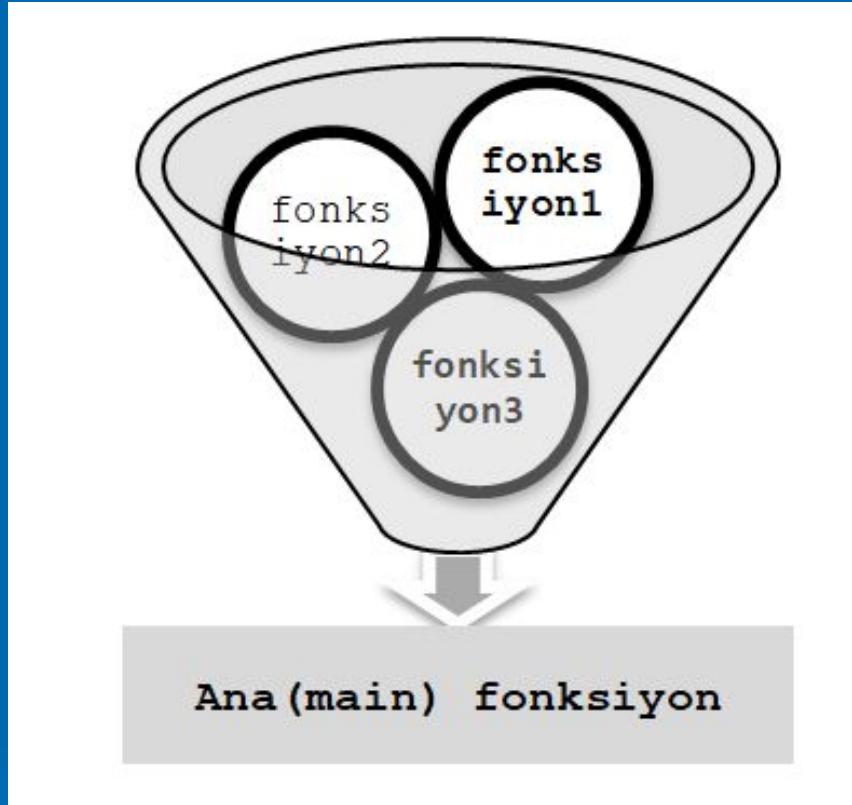
FONKSİYONLAR

C, Fonksiyonel bir dildir.

C++, Hem fonksiyonel hem de nesne yönelimli bir dildir.

- C dili fonksiyonlardan oluşan bir dildir.
- Her C programı en azından bir **main ()** fonksiyonu içerir.
- Ana programı (**main** fonksiyonu) oluşturan her bir alt program, **fonksiyon** (function) ismini alır.

Fonksiyonel bir dil C



- Bir ana (main) program birden fazla fonksiyonun (alt programın) birleşiminden oluşabilir.

Fonksiyonların genel özelliklerি;

- Fonksiyonlar, bir defa oluşturulur, birden fazla kez çağrılabılır.
- Başka programlar tarafından çağrırlar.
- Fonksiyonların girdilerine **parametre** veya **argüman** adı verilir.
- Parametre geçişine (fonksiyon dışından değer alımına) izin verirler.
- Bir fonksiyon ana programda sadece ismi belirtilerek çağrılabılır.
- Fonksiyonlar değişkenlere benzerler. Değişkenler bir değer içerirken **fonksiyonlar bir sonuç üretir**. Nasıl ki değişkenlerin veri tipini tanımlamak zorunda ise benzer şekilde **fonksiyonların da veri tiplerini tanımlamak zorundayız**.

Fonksiyonların genel özelliklerı;

- C/C++ dillerinde fonksiyon içerisinde fonksiyon tanımlanmaz.

Hatalı fonksiyon tanımı	Doğru kullanımı;
<pre>void Kitap() /*Fonksiyon-1*/ { cout<< ("Kitap: C/C++\n"); void Yazar() /*Fonksiyon-2*/ { /*Hatalı tanımlama*/ cout<< ("Yazar:B. Çobanoğlu\n"); } }</pre>	<pre>void Yazar() /*Fonksiyon-1*/ { cout<< ("Yazar:B.Çobanoğlu\n"); } void Kitap() /*Fonksiyon-2*/ { cout<< ("Kitap: C/C++\n"); Yazar(); }</pre>

Fonksiyon Tanımlama: void tipli fonksiyonlar

Parametreli void (geriye değer döndürmeyecek) fonksiyon

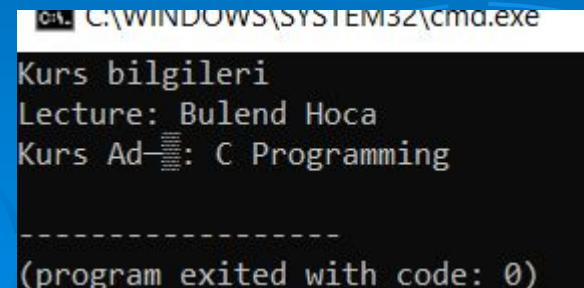
```
void fonk_Adi (arg1, arg2)
{
    ....;
    ....;
}
```

Parametresiz void fonksiyon

```
void fonk_Adi( )
{
    ....;
    ....;
}
```

```
1 #include <stdio.h>
2 //Kurs isimli fonksiyon
3 void Kurs()
4 {
5     printf ("Lecture: Bülend Hoca\n");
6     printf ("Kurs Adı: C Programming");
7 }return;
8 //Ana program
9 int main()
10 {
11     printf ("Kurs bilgileri\n");
12     Kurs(); // Kurs isimli fonk. çağrıldı
13     return 0;
14 }
15 }
```

Kurs() fonksiyonu
Geri dönüş değerine de
parametreye de sahip değildir.



```
C:\WINDOWS\SYSTEM32\cmd.exe
Kurs bilgileri
Lecture: Bülend Hoca
Kurs Adı: C Programming
-----
(program exited with code: 0)
```

Parametreli void tipli fonksiyonlar

```
1 #include<stdio.h>
2 //Parametreli void Fonksiyon
3 void show (int mark){
4     printf("%d\n", mark);
5 }
6 //Ana program
7 int main() {
8     int m[]={10, 20, 30, 40, 50};
9     int i;
10    for(i=0; i< 5; i++)
11        show(m[i]);
12    return 0;
13 }
```

Fonksiyonlara parametre aktarımı

```
void foo(int a, b) /* geçersiz! */  
{  
    /* ... */  
}
```

```
void foo(int a, int b) /* geçerli */  
{  
    /* ... */  
}
```

Parametreli bir fonksiyon çağrılrken parametre sayısı kadar argüman kullanılır. Örneğin:
foo(10, 20);

Geri Dönüş Değeri Olan Fonksiyonlar

- Geri dönüş tipi olarak, değişken tanımlamada kullanılan herhangi bir veri tipi {int, double, float,...} kullanılabilir.

Parametreli geriye değer döndüren fonksiyon	Parametresiz geriye değer döndüren fonksiyon
dönüşTipi fonk_Adı (arg1, arg2) { <u>int</u>; return <u>dönüşDeğeri</u> ; } <u>int</u>	dönüşTipi fonk_Adı() {; return dönüşDeğeri; }

- Fonksiyon dönüş tipi ile return komutu yanında yazılan değer tipinin aynı olması gerektiğini unutmayın.

Geri Dönüş Değeri Olan Fonksiyonlar

Soru: Bu programın çıktısı ne olur?

```
1 #include<stdio.h>
2 int tic (int x) {
3     if (x==10)
4         return x;
5     else
6         return x + 2;
7 }
8
9 int main () {
10    int a = 13;
11    a = tic(a);
12    printf ("%d", a);
13    return 0;
14 }
15 }
```

Çözüm:

Değişken takibi yaparak
Adım adım ilerlemektir.

Return komutu

Komut örneği	Dönüş değeri	Açıklama
return; <i>//void fonksiyonlarda kullanılabilir</i>	-	Bu örnekte fonksiyonun geri dönüş değeri yoktur, sadece fonksiyonu sonlandırır.
return 7;	7	Bu örnekte fonksiyonun geri dönüş değeri sabit bir sayıdır
return a++;	a+1	Bu örnekte fonksiyonun geri dönüş değeri bir değişkendir
return (a%2) ? 0: 1;	0 ya da 1	Bu örnekte a'nın 2 ye bölümünden kalan değer geri döndürülür.
return fakt();	fakt() fonksiyonu sonucu	Bu örnekte fonksiyonun geri dönüş değeri, başka bir fonksiyonun “fakt()” geri dönüş değeridir.

Fonksiyon Prototipi

- C/C++ derleyicisi (compiler) kod derleme işlemini yukarıdan aşağıya (1. Satırdan, sonuncu satıra) doğru yapar.
- Dolayısıyla eğer çağrılan **fonksiyon**, ana (main) veya çağrıran fonksiyondan sonra düzenlenecekse öncesinde **fonksiyon prototipinin/şablonunun** aşağıdaki gibi tanımlanması gereklidir;
 - <Tip> **fonksiyonAdı** <parametre tipi>;
`int foo (int);`

Fonksiyon Prototipi

```
1 #include<stdio.h>
2 //toupper() ilevi gören fonksiyon
3 char capitalize(char c) {
4     if (c >= 'a' && c <= 'z')
5         return (c - ('a' - 'A'));
6     return c;
7 }
8 //Ana program
9 int main () {
10    char c='D';
11    printf ("%c",capitalize(c));
12    return 0;
13 }
```

```
1 #include<stdio.h>
2 //Ana program
3 int main () {
4     char c='D';
5     printf ("%c",capitalize(c));
6     return 0;
7 }
8 //toupper() islevi gören fonksiyon
9 char capitalize(char c) {
10     if (c >= 'a' && c <= 'z')
11         return (c - ('a' - 'A'));
12     return c;
13 }
```

```
arfCevir.c" (C:\Users\bc\Desktop dizininde)
function 'main':
5: warning: implicit declaration of function 'capitalize' [-Wimplicit]
alize(c));
~~~~~
top level:
: error: conflicting types for 'capitalize'
```

Fonksiyon Prototipi

```
1 #include<stdio.h>
2 //Ana programda cagrılan fonk. prototipi
3 char capitalize(char);
4 //Ana program
5 int main () {
6     char c='d';
7     printf ("%c",capitalize(c));
8     return 0;
9 }
10 //toupper() islevi gören fonksiyon
11 char capitalize(char c) {
12     if (c >= 'a' && c <= 'z')
13         return (c - ('a' - 'A'));
14     return c;
15 }
```

Fonksiyonlara parametre olarak dizi aktarımı

- Dikkat edilirse; fonksiyon tanımlanırken eğer parametresi dizi ise köşeli parantezler ‘[]’ belirtilirken,
- aynı fonksiyon çağrılrken bu köşeli parantezler belirtilmmez.
- Ayrıca fonksiyon parametresinin dizi olduğu '*' işaretçi(pointer) operatörü ile de belirtilebilir.

Fonksiyon Tanımlanması	Örnek Kullanımı
Tip fonksiyonAdi (tip diziAdi[],) veya	void notlariAl(int Nt[]) veya
Tip fonksiyonAdi (tip *diziAdi,)	void notlariAl(int *Nt)
Fonksiyonun Çağrılması	Örnek Kullanımı
fonksiyonAdi (diziAdi)	notlariAl(progNot);
Not. Fonksiyon çağrılrken [] parantezlerin kullanılmadığına dikkat ediniz...	

#define ile fonksiyon tanımlama

Sayının işaretini veren signum() isimli bir fonksiyon tasarlayalım...

1. Formu:

```
int signum(int X)
{
    return
    (X<0 ? -1 : X>0 ? +1: 0);
}
```

function

Fark Nedir?

2. Formu:

makro

```
#define signum(X) (X<0 ? -1 : (X>0 ? +1: 0))
```

Makro mu? Fonksiyon mu?

```
#define KARE(a) (a*a)
```

Tek satırlı küçük fonksiyonların fonksiyon olarak değil de makro olarak organize edilmesi daha uygun olur.
Büyük kodların makro olarak tanımlanması kodu ciddi biçimde büyütебilmektedir.

O halde çok küçük fonksiyonlar makro olarak diğerleri normal fonksiyon olarak yazılabilirler

Kendi strcmp fonksiyonumuzun tasarıımı

strcmp() fonk. çalışma mantığı;

- int strcmp (char *s1, char *s2);
- s1>s2 ise +
- s1<s2 ise -
- s1 == s2 ise 0 sonucunu üretir. Buna göre

```
//for eşdeğeri;
```

```
int i;
for ( i=0; toupper(s1[i]) == toupper(s2[i]);++i) {
    if (s1[i] == '\0')
        return 0;
}
return s1[i] - s2[i];
```

Kendi string fonksiyonumuzu yazalım.

Soru: Kendi strlen() fonksiyonumuzu yazalım

Çözüm.

strlen fonksiyonu bir string ifadenin karakter uzunluğunu bulmak için kullanılıyor idi.

```
#include <stdio.h>
#include <string.h>
unsigned int my_strlen(char *str)
{
    int i;
    // döngü değişkeni karakter sayacı olarak kullanılabilir.
    for (i = 0; str[i] != '\0'; ++i);
    return i;
}
//main fonksiyonu
int main()
{
    printf("%u", my_strlen("Bülend Hoca"));
}
```

Özyinelemeli-Rekürsif Fonksiyonlar (Recursive Functions)

- Kendini doğrudan veya dolaylı olarak çağrıran fonksiyonlara **özyinelemeli /rekursif (recursive) fonksiyonlar** adı verilir.
- **Özyineleme**, en genel anlamıyla bir yapının kendi kendini çağrımasıdır.
- Özellikle matematiksel (bileşke fonksiyonlarda) işlemlerde sıkılıkla kullanılır.
- Mesela ünlü Fibonacci serisi: 1,1,2,3,5,8,13... şeklinde gider. Serinin her elemanı kendisinden bir ve iki önceki elemanların toplamı ile oluşur. $F(n) = F(n - 1) + F(n - 2)$
- Özyineleme (Recursion); algoritma tasarımını basitleştirir ancak tekrarlı yapılara göre fonksiyon çağrıma sayısı arttığı için bellek alanı da artar.

Yaşamdan rekürsif örnekler



- Çam kozalaklarının oluşması,
- Bir ağacın büyümesi sırasında büyüyen dal ve yaprakların bir önceki seneki yapıların üzerinde yükselmesi,
- Matruşka
-



Özyinelemeli-Rekürsif Fonksiyonlar

Soru-Cevaplar

- main () fonksiyonu; özyinelemeli f () fonksyonu çağrılığında;
 - genelde o fonksiyonu kaç kez yürütüleceğini/çalıştıracağını bilemez.
- Bir fonksiyonun Özyinelemeli (rekürsif) fonksiyon olduğu nasıl anlaşılır?

Faktoriyel Alan Normal Fonksiyon

```
int factorial(int n) {  
    int i, j = 1;  
    for (i = 1; i <= n; i++)  
        j *= i;  
    return j;  
}
```

$$n! = 1 * 2 * 3 * 4 * 5 * 6 * \dots * (n-1) * n$$

Özyinelemeli-Rekürsif Fonksiyon Örneği: Faktöriyel Alma

$$6! = 6 * 5! = 6 * 5 * 4! = 6 * 5 * 4 * 3! = 6 * 5 * 4 * 3 * 2! = 6 * 5 * 4 * 3 * 2 * 1 = 720$$



5*4!



4*3!



3*2!



2*1!

$$n! = n * (n-1)!$$

```
int faktoriyel(int n)
{
    if (n <= 1) return n;
    else return (n * faktoriyel(n-1));
}
```

Algoritma: OBEB (Özyinelemeli-Rekürsif Fonksiyon Örneği)

En eski algoritmalarlardan biri #öklit algoritmasıdır. Bu algoritma #OBEB problemini çözer.

İşte bu #gcd (obeb) algoritmasının C/C++ kodlanması;

```
int gcd(int a, int b) {  
    if (a == 0)  
        return b;  
    return gcd(b%a, a);  
}
```

<u>a</u>	<u>b</u>	<u>b%a</u>
12	8	8 (kalan)
8	12	4
4	8	0
0	4	

Özyinelemeli-Rekürsif Fonksiyon Örneği: Çıktı sorusu

```
#include <stdio.h>
int tic (int x) {
    if (x==10) return x;
    else return x * 2;
}
int tac (int x) {
    if (x==10) return x;
    else return x - tac (x - 2);
}
int main () {
int a = 18;
    a = tic(tac(a));
    printf ("%d", a);
    return 0;
}
```

```
//tic(tac(18));
//tac(18) = 18- tac(16) = 18-16-tac(14)
//18-16-14-tac(12)=18-16-14-12-tac(10) =14
//tic(14) = 14*2 = 28
```

TASK: Rekürsif Fonksiyon Çıktı Sorusu



```
#include <stdio.h>
int fun (int a) {
    if (a == 0) return 0;
    else return 1 + fun(a + 1);
}
int main () {
    printf ("%d", fun(23));
    return 0;
}
```

??



TASK: Rekürsif Fonksiyon Çıktı Sorusu

```
#include <stdio.h>
int tic (int x) {
    if (x==10) return x;
    else return x + 2;
}
int tac (int x) {
    if (x==10) return x;
    else return x - tac (x - 1);
}
int main () {
int a = 13;
    a = tac(tic(a));
    printf ("%d", a);
    return 0;
}
```

??