FONKSİYONLAR

```
[geri dönüş değeri] <fonksiyon ismi>([paramatre])
{
    ANABLOK
}
```

Her fonksiyon ard arda tanımlanır. İç içe fonksiyon tanımlanamaz...

Yanlış fonksiyon kullanımına örnek:

```
 main()
{
fonk()
{   }
}
```

Olması gereken:

```
 main()
{
}
fonk()
{
}
```

Hiçbir fonksiyon 1'den fazla tanımlanamaz. En azından aynı isme sahip olmaz.

```
#include <stdio.h>
int fonk1();
int fonk2();
int fonk3();
main()
{       printf("Ben main'im\n");
        fonk1();
        fonk2();        }

int fonk1()
{       printf("Ben 1.fonksiyonum\n");       }

int fonk2()
{       printf("Ben 2. fonksiyonum\n"); fonk3();    }

int fonk3()
{       printf("Ben 3. fonksiyonum\n");      }
```

FONKSİYONLARIN GERİ DÖNÜŞ DEĞERLERİ(RETURN VALUE)

```
int x;

x = fonk();
```

Bir fonksiyonun çalışması bittikten sonra onu çağıran fonksiyona gönderdii değere geri dönüş değeri denir.

Fonksiyonların geri dönüş değerleri aritmetik işlemlere sokulabilir.

```
x = fonk() + a;
```
gibi..

GERİ DÖNÜŞ DEĞERİ OLUŞTURULMASI VE return ANAHTAR SÖZCÜĞÜ

```
#include <stdio.h>
int fonk()
{
        printf("Ben fonk'um\n");
        return 100;          }

main()
{
        int a;
        a = fonk();
        printf("fonk'un geri dönüş değeri=%d\n",a);      }
```

return anahtar sözcüğünün iki işlevi vardır:
1-Fonksiyonun çalışmasını bitirir. Bu durumda akış, onu çalıştıran fonksiyonda devam edecektir.
2-Geri dönüş değeri oluşturur.

Kullanım biçimi => return [ifade]

return İFADESİ NASIL OLUŞTURULUR?

return'ün yanındaki ifade önce derleyici tarafından programcının erişemediği geçici bir bölgeye alınır ve oradan kullanılır. Örneğin a = fonk(); işleminde şunlar yapılır:
temp = fonk();

a = temp; Bu işlem bize aksettirilmez.

Fonksiyonun geri dönüş türü aslında geçici bölgenin türünü gösterir. return ifadesinin oluşturulması da geçici bölgeye yapılan gizli bir atamadır.

```
long fonk()
{
        printf("Ben fonk'um\");
        return 123000L;
}

main()
{
        long a;
        a = fonk();
        printf("%ld\n",a);
}
```

return anahtar sözcüğü kullanılmışsa fonksiyon ana bloğu bittiğinde sonlanır. Fonksiyonda return ile belirli bir değer verilmemişse rastgele bir geri dönüş değeri verilecektir.

Bir fonksiyonun geri dönüş değerine sahip olması kullanılmasını gerektirmez.

Fonksiyon başında void yazılırsa foksiyonun geri dönüş değerinin olmadığı analtılır.

```
void fonk()
{
 }
```

Böyle fonksiyonlarda return anahtar sözcüğü fonksiyonu sonlandırmak için kullanılır.

```
#include <stdio.h>

void fonk()
{           printf("selam\n");
            return;    }

main()
{           int a;
a = fonk(); /*bu kullanım tamamen yanlış*/
printf("%d\n",a); /*bu kullanım tamamen yanlış*/
}
```

NESNELERİN FAALİYET ALANLARI VE ÖMÜRLERİ

Faaliyet alanları:

Bir nesnenin derleyici tarafından tanınabildiği program aralığını gösterir. 3 tür faaliyet alanı vardı.

1-Blok faaliyet alanı:Yalnızca bir blokta tanınır.
2-Fonksiyon faaliyet alanı:Bir fonksiyonun her yerinde tanınır.
3-Dosya faaliyet alanı:Programın her tarafında tanınır.

NESNELERİN FAALİYET ALANLARINA GÖRE SINIFLANDIRILMASI

1-Yerel değişkenler:

Blokların başlarında tanımlanmış değişkenler yerel değişkenlerdir. Bunlar tanımlandıkları blokta kullanılırlar.(blok faaliyet alanı)

```
void main(void)
{
int a;
        {
        int b;
        }
}
```

Burda a main fonksiyonunun tamamında b ise sadeec içteki blok'ta geçerlidir.Farklı faaliyet alanlarına sahip değişkenler aynı ada sahip olabilir.

2-Global değişkenler

Tüm blokların dışında tanımlanan değişkenlere global değişkenler denir. Dosya faaliyet alanı kuralına uyar. Kaynak kodun her yerinde tanınır.

```
#include <stdio.h>
int a;

void fonk(void)
{               a = 20; }

main()
{
        a = 10;
        fonk();
        printf("%d\n", a);/*ekrana 20 değeri basılır*/
}
```

3-Parametre değişkenleri

Bir fonksiyon parametre alabilir. Bunun için parametre değişkenlerinin tanımlanması gerekir. İki ayrı yöntemsel parametre bildirimi yapılır.

a.Eski biçim:

```
void fonk(a,b)
int a;
long b;
{
 }
```

b.Yeni biçim

```
void fonk(int a,long b)
{
 }
```

Bu biçimde parametre değişkenler aralarına virgül konularak tanımlanır.

```
void fonk(int a,b)/*YANLIŞ TANIMLAMA*/
```

Bu değişkenler fonksiyon faaliyet alanı kuralına uyarlar. Parametre değişkenine sahip bir fonksiyon aynı sayıda değişkenle çağırılmalıdır.

PARAMETRE AKTARIM KURALI

Parametreli bir fonksiyon çağırıldığında derleyici önce parametrelerden parametre değerlerine karşılık atama yapar, daha sonra programın akışı fonksiyona geçirilir.

```
#include <stdio.h>

int add(int a,int b)
{
	return a + b;
}

main()
{
	int x = 10, y = 20, z;

	z = add(x, y);
	printf("%d\n", z);
}
```

Fonksiyonlar sabitlerle de çağırılabilirler.
fonk(10, 20); gibi..

math.h	Matematiksel Fonksiyonlar
stdlib.h	Standart başlık dosyası bazı fonksiyonlar
ctype.h	Karakter Üzerinde İşlem Yapan Fonksiyonlar

math.h Matematiksel Fonksiyonlar

Matematiksel fonksiyonlarında kullanılan değişken tipleri genelde double değişkenlerdir.

Bu fonksiyonlardan biri program içinde kullanılacaksa **#include<math.h>** başlık dosyası program içine eklenmelidir.

En çok kullanılan matematiksel fonksiyonlar şu şekildedir.

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
int abs(int x);	x tamsayısının mutlak değerini hesaplar	abs(-4)	4
double fabs(double x);	x gerçel sayısının mutlak değerini hesaplar	fabs(-4.0)	4.000000
double floor(double x);	x'e (x'den küçük) en yakın tamsayıyı gönderir	floor(-2.7)	-3.000000
double ceil(double x);	x'e (x'den büyük) en yakın tamsayıyı gönderir	ceil(5.6)	6.000000
double sqrt(double x);	pozitif x sayısının karekökünü hesaplar	sqrt(4.0)	2.000000
double pow(double x, double y);	x^y (x^y) değerini hesaplar	pow(2.0,3.0)	8.000000
double log(double x);	pozitif x sayısının doğal logaritmasını hesaplar, ln(x)	log(4.0)	1.386294
double log10(double x);	pozitif x sayısının 10 tabanındaki logaritmasını hesaplar	log10(4.0)	0.602060
double sin(double x);	radyan cinsinden girilien x sayısının sinüs değerini hesaplar	sin(3.14)	0.001593
double cos(double x);	radyan cinsinden girilien x sayısının kosinüs değerini hesaplar	cos(3.14)	-0.999999
double tan(double x);	radyan cinsinden girilien x sayısının tanjant değerini hesaplar	tan(3.14)	-0.001593
double asin(double x);	sinüs değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır.	asin(0.5)	0.523599
double acos(double x);	cosinüs değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır.	acos(0.5)	1.047198
double atan(double x);	tanjant değeri x olan açıyı gönderir. Açı -pi/2 ile pi/2 arasındadır.	atan(0.5)	0.463648

```
/* yuvarlamalar örneği */

#include <math.h>
#include <iostream>
using namespace std;
int main ()
{
  double x=3.57;
  cout<<floor(x)<<endl; //aşağı yuvarla
  cout<<ceil(x)<<endl; //yukarı yuvarla
  cout<<round(x)<<endl; //enyakın tamsayıya yuvarla
  return 0;
}
```

```
/* sqrt örneği */
#include <iostream>
#include <stdlib.h>
#include <math.h>

using namespace std;
int main()
{
    float ans;
    ans = sqrt(25.0);
    cout << "25 sayisinin karekoku: "<<ans<< endl;
    system("PAUSE");
    return 0;
}
```

```
/* pow örneği */
#include <stdio.h>      /* printf */
#include <math.h>       /* pow */
#include <iostream>
using namespace std;
int main ()
{
  printf ("7 ^ 3 = %f\n", pow (7.0, 3.0) );
  printf ("4.73 ^ 12 = %f\n", pow (4.73, 12.0) );
  printf ("32.01 ^ 1.54 = %f\n\n", pow (32.01, 1.54) );

  //aynı işlemleri cout ile yapalım
  cout<<"7 ^ 3 ="<<pow (7.0, 3.0)<<endl;
  cout<<"4.73 ^ 12 ="<<pow (4.73, 12.0)<<endl;
  cout<<"32.01 ^ 1.54 ="<<pow (32.01, 1.54)<<endl;
  return 0;
}
```

Trigonometrik (sin, cos, tan) fonksiyonlar kendisine parametre olarak gelen değeri radyan olarak kabul eder ve sonucu hesaplar. Eğer açılar derece cinsinden hesaplanması gerekiyorsa şu dönüşüm kullanılmalıdır:

radyan = (3.141593/180.0)*derece;

Sin(60)	60 Radyan derecenin sinüsü
Sin(pi/180*60)	60 Gradyan derecenin sinüsü

Örnek: *sin(), cos(), and tan() fonksiyonlarını kullanan bir program yapalım*

```c
#include <stdio.h>
#include <math.h>
#define PI     3.141593
#define CARPAN PI/180.0

main()
{
double x;
x  = 30.0; /* 30 derece !   */
x *= CARPAN; /* radyana çevir */
printf("30 derecenin sinusu   : %f\n", sin(x));
printf("30 derecenin kosinusu : %f\n", cos(x));
printf("30 derecenin tanjanti : %f\n", tan(x));
}
```

STANDART FONKSİYONLAR(stdlib.h)

Programlarınızda kullanılmak üzere bir dizi fonksiyon stdlib.h başlık dosyasında tanımlanmıştır. Bu dosya C++ ile birlikte gelmektedir. İçerisinde bir takım bulunan hazır fonksiyonlar ile algoritmalar güçlendirilebilir. İçerisindeki fonksiyon yapıları şunlardır.

1. Sonlu string bilgiyi sayısal değer dönüştürme
2. String bilginin sonunu belirleyerek sayısal değere dönüştürme
3. Program sonlandırma
4. Hafıza yönetimi
5. Rasgele sayılar
6. Tam sayılarla ilgili fonksiyonlar

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
int atoi(const char *s);	Bir katarı tamsayıya çevirir	atoi("-12345")	-12345
long atol(const char *s);	Bir katarı uzun tamsayıya çevirir	atol("1234567890")	1234567890
double atof(const char *s);	Bir katarı reel sayıya çevirir	atof("-123.546")	-123.456
void exit(int durum);	Programı sonlandırarak kontrolü işletim sistemine geri verir.	exit(0)	-
int rand(void);	0 ile RAND_MAX arasında rastgele sayı üretir. RAND_MAX, stdlib.h içinde tanımlanmış bir sembolik sabittir	rand()	504851
max(a,b)	stdlib.h'de tanımlanmış iki sayıdan en büyüğünü bulan makro fonksiyon	max(5,9)	9
min(a,b)	stdlib.h'de tanımlanmış iki sayıdan en küçüğünü bulan makro fonksiyon	min(5,9)	5
void *malloc(unsigned boyut);	Bellekte boyut ile belirtilen sayıda bayt kadar yer tahsis eder	p=malloc(sizeof(int)*5)	?
void free(void *ptr);	malloc ile ayrılan bellek bölgesini boşaltır	free(p)	-

Sonlu string bilgiyi sayısal değer dönüştürme

Bu fonksiyonların hepsinin yapısı birbirine benzediği için anlaşılması bir hayli kolay. Hepsi yapı olarak birbirine benziyor.

DonusVeriTipi atox (*char) ;Yukardaki yapıda X bizim ne tür veri istediğimizi gösteriyor; örnegin fload ise burası f olacak.o durumda DönüşVeriTipi de fload olacaktır. Bu tür fonksiyonlara birkaç örnek:

```
float   atof(*char);
int     atoi(*char);
double  atod(*char);
long    atol(*char);
```

```c
//ÖRNEK PROGRAM
#include<stdio.h>
#include<stdlib.h>
int main() {
char tam_bir_sayi[] = "156";
char yas[] = "18 yaşındayım";
printf( "%d\n", atoi( tam_bir_sayi ) );
printf( "%d\n", atoi( yas ) );
printf("ornek carpim:%d\n",atoi(tam_bir_sayi)*atoi(yas));
return 0;
}
```

String bilginin sonunu belirleyerek sayısal değere dönüştürme

Bu fonksiyonlar ise bir karakter kümesi içinden belirli bir aralığı sayısal veriye dönüştürür.Kullanımı sonlu string bilgiyi sayısal değer dönüştürme fonksiyonlarına benzese de farklılık içerir ve biraz karışıktır.

DönüsVeriTipi strtoX (*char,**char,int);Yine x dönüştürlücek veri tipini gösteriyor burası d(double) l(long) ve ul (unsigned long) isimlerini alabiliyor. ilk *char değişkeni bize dönüştüreceğimiz stringin başlangıç noktasını veriyor. İkinci argüman ise asıl karışık nokta. Dönüştürmeyi bitireceğimiz noktanın pointer adresi.

```c
//ÖRNEK PROGRAM
#include<stdio.h>
#include<stdlib.h>
main()
{
char *strdegisken = "135.57abc",*son;
double sonuc;
sonuc = strtod(strdegisken,&son); //string ifade double dönüştürülüyor
printf ("Donusumm sonucunda deger  %f  oldu\n",sonuc);
printf ("Kalan kisim:%s\n",son);
printf("***************\n");
return 0;
}
```

Program sonlandırma

Normalde main program alt programları cağırır ve main programda return satırına ulaşıldığında veya "}" karakteri ile karşılaşıldığında main uygulama son bulmuştur denir. Ama standart kütüphane bize üç farklı fonksiyon ile program sonlandırma imkanı da verir.

abort

Parametre almadan kullanılır. Programı sonlandırır. Sonlandırmakla da kalmaz açılmış dosyaları kapatır ve bufferi temizler. Böylece tek tek dosyaları kapatıp belleği geri teslim etmezsiniz.

atexit

Esasında programı sonlandıran bir komut değildir. Fakat ana program sonlandığında işletilecek programı verir. 32 adet programı sıralar ve en son girilen program önce çalışır.

exit

Programı bir değer ile sonlandırır. Linux altında sık kullanılan bir yapıdır.

```c
//ÖRNEK
#include <stdio.h>
#include <stdlib.h>
void cikis1 (void)
{
puts ("cikis1");
}
void cikis2 (void)
{
puts ("cikis2");
}

int main (){
atexit (cikis1);
atexit (cikis2);
puts ("main fonksiyon.");
return 0;
}
```

system( ) kullanımı

System fonksiyonu sistem komutlarını yürütmek için kullanılır.Bu fonksiyon kendisine parametre olarak gelen ifadeyi UNIX, Linux veya MS-DOS komut satırına yazar ve çalıştırır. system() fonksiyonu ile, bilgisayarın tüm dosya ve çevre birimleri, küçük program parçaları sayesinde kontrol edilebilir. Genel yazım biçimi:

```
system("işletim_sistemi_komutu");
```

Örneğin,

Windows dizininde bulunan tüm dosyaları listelemek için

```
system("dir c:\windows");
```

Bu fonksiyon Linux İşletim sisteminde de kullanımı aynıdır.

```c
//ÖRNEK
#include <stdio.h>
#include <stdlib.h>
int main ()
{
int i;
printf ("Sistem kontrol edilebilir durumdamı:...");
if (system(NULL)) puts ("Ok");
else exit (1);
printf ("dir komutu calistiriliyor\n");
i=system ("dir");  /*DIR KOMUTU ÇALIŞTIRILIYOR*/
printf ("Geri donus degeri: %d.\n",i);
return 0;
}
```

rand() fonksiyonu kullanımı

```
#include <stdio.h>
#include <cstdlib>
#include <conio.h>
#include <time.h> //time(NULL)
main() {
int   i,ri;
float rf;
char x;
srand(time(NULL));//her seferinde rastgele sayı uretimi için
basla:
system("cls");
for(i=0;i<10;i++) {
ri = rand() % 100;              /* 0-100 arası tamsayı   */
rf = (float) rand()/RAND_MAX;  /* 0-1   arası reel sayı */
printf("%d\t%f\n",ri,rf);
}
puts("\n20 tane rasgele sayi urettim...\n");
puts("Devam etmek istiyorsanız e harfine basınız...");
x=getch();
if(x=='e'||x=='E') goto basla;  }
```

Hafıza yönetimi

C dilinde yazılan bir program bilgisayarda kullandığı belleği bazı parçalara ayırarak kullanır. Windows XP altında çalışan programlarda toplamda 4 GB olan bu bellek bazı bölümlere ayrılır. Program bölümü, kalıcı değişken bölümü, yerel değişkenlerin kullandığı yığıt ve dinamik hafıza ayırma işlemlerini yaptığımız yığın. İşletim sisteminden yararlanarak sistemden yer isteme işlemine hafızada yer açma (memory allocation) denir.

Bunlar, hafızada yer açmamızı sağlayacak olan malloc( ) fonksiyonu ve işimiz bittiğinde kullandığımız hafızayı iade etmemizi sağlayacak olan free( ) fonksiyonudur. Bunların dışında bu kütüphanede calloc( ) ve realloc( ) fonksiyonları da tanımlanmıştır. Bu fonksiyonları pointer larla kullanacağımızı unutmayalım. Şimdi bu fonksiyonlara bir bakalım:

Malloc

Yığında yer açmamızı sağlayacak olan malloc( ) fonksiyonunun genel prototipi şöyledir:

```
void * malloc(int size)
```

Burada size ile ifade edilen yere istenen byte sayısı girilir. Byte sayısını hatalı girebiliriz bu yüzden burada sizeof( ) fonksiyonu kullanılmalıdır. Hem bu sayede bir int veya float tipinin kaç byte yer ayırdığını aklımızda tutmak zorunda kalmayız. Bu yüzden bırakalım da bu konuyu bize hazır sunulan sizeof( ) fonksiyonu düşünsün.

Not: malloc( ) fonksiyonu default olarak void bir değer döndürür. Bu, ayrılan hafızanın başlangıç adresidir. Bu yüzden bu fonksiyonu her kullandığımızda gerçekten bu istediğimiz yer ayrıldı mı diye kontrol etmek gerekir. Eğer fonksiyon başarılı olamamışsa NULL değer döndürür.

ÖRNEK

```
int *p;
p= malloc( sizeof(int) );
...
```

Dinamik hafıza ayırma fonksiyonlarını pointerlarla kullanacağımızı söylemiştik. Bunun için önce bir p pointerı tanımladık ve mallocla ayırdığımız yeri bu p pointerına döndürdük.

FREE

Program sona erdiğinde ayırdığımız hafızayı yığına iade etmemiz gerekir. C dili kullanıldıktan sonra iade edilmeyen hafızayı "çöp" olarak nitelendirir. Bu çöpü bizim toplamamız gerekir. Bunun için free( ) fonksiyonu kullanılır. Prototipi şöyledir:

```
void free( void *p);
```

Örneğin deminki örnekten yola çıkarsak, malloc( ) ile açtığımız hafızayı p pointerına döndürmüştük. İşimiz bittiğinde free( ) fonksiyonunu kullanarak açtığımız bu yeri iade edeceğiz. Bunun için,

```
int *p;
p= (int*) malloc( sizeof(int) );
   ...
   free(p);
```

yazmamız yeterlidir.

Calloc

calloc( ) fonksiyonu aynı malloc( ) gibi hafızada yer açar ve bu açılan bytelara 0 değerini atar. Prototpi şöyledir:

void *calloc(int sayı, boyut);

```
//ÖRNEK PROGRAM
#include <stdio.h>
#include <stdlib.h>
#define YER 1000
 main() {
int *p, i;
/* YER=1000 boyutunda ve int türünde yer acalım */
p = (int *) malloc(YER * sizeof(int));
/* artık p yi 1000 boyutunda bir dizi olarak kullanabiliriz */
for (i=0; i < YER; i++)
{ /* bu dizi elemanlarına rastgele degerler atayalım */
p[i] = rand();
}  /* isimiz bitti simdi tesekkur edelim ve hafızayı iade edelim */
free(p);   }
```

Realloc

Çoğunlukla bir bellek bloğunu kullanmaya başlarken ihtiyacınız olacak bellek miktarını bilemez ve yaklaşık bir boyut ile bloğu ayırırsınız. Örneğin, blok bir dosyadan okunan satırı tutan bir tampon olabilir ve bir satır için yeterli olan tamponunuz başka bir satır için yetersiz kalabilir.Bunu değiştirmek için realloc kullanılır.
Çoğu durumda realloc başarısız olduğunda eski bloğa ne olduğunun fazla bir önemi kalmaz. Çünkü uygulama yazılımı bellek yetmediğinde çalışmasını sürdüremez ve yapacak tek şey kalır: bir ölümcül hata iletisi ile kendini sonlandırmak. Çoğunlukla bu durum için bir yordam yazılır. realloc için xrealloc dur.

KULLANIM ŞEKLİ

```
* xrealloc (void *ptr, size_t size)

{
  char *str =realloc (ptr, size);
  if (str == 0)
    printf ("Sanal bellek tükendi");
  return o;
}
```

TAM SAYI İLE İLGİLİ FONKSİYONLAR

ABS

Bir tamsayının mutlak değerini hesaplar.

```
#include <stdio.h>
#include <stdlib.h>
int main (){
int n,m;
n=abs(23);
m=abs(-11);
printf ("n=%d\n",n);
printf ("m=%d\n",m);
return 0;}
```

ÇIKTI

n=23

m=11

DİV

Birinci parametreyi ikinci parametreye böler, bölümü ve kalanı div_t yapı türünde döndürür.

```
#include <stdio.h>
#include <stdlib.h>
int main (){
div_t divresult;
divresult = div (38,5);
printf ("38 div 5 => %d, KALAN %d.\n", divresult.quot, divresult.rem);
return 0;
}
```

ÇIKTI

38 div 5 => 7, KALAN 3.

LABS

```
#include <stdio.h>
#include <stdlib.h>
int main (){
long int n,m;
n=labs(65537L);
m=labs(-100000L);
printf ("n=%ld\n",n);
printf ("m=%ld\n",m);  }
```

ÇIKTI

```
n=65537
m=100000
```

LDİV

```
#include <stdio.h>
#include <stdlib.h>
int main (){
ldiv_t  ldivresult;
ldivresult = ldiv (1000000L,132L);
printf ("1000000 div 132 => %ld, KALAN %ld.\n", ldivresult.quot, ldivresult.rem);  }
```

ÇIKTI

```
1000000 div 132 => 7575, KALAN 100.
```

KARAKTER FONKSİYONLARI(ctype.h)

Bu başlık doyası kabaca C dilinde bulunan karakterler (char) üzerinde işlem yapmaya imkan sağlar. Başlık dosyası zaten ismi de buradan almıştır (char + type = ctype). Amacı bir karakterin tipini algılama veya değiştirmektir. Kütüphane karakterler üzerinde işlem imkanı sağladığı için dizgiler (string)üzerinde yapılan işlemler açısından avantaj sağlar.

Kütüphane herhangi bir C kodunda aşağıdaki şekilde eklenir.

```
#include <ctype.h>
```

Eklemenin ardından aşağıdaki fonksiyonlar çağırılabilir:

Komut	Açıklama	Örnek	Sonuç
isalpha(c)	c bir harf ise 0 dan farklı, değilse 0 gönderir	isalpha('a')	8
isalnum(c)	c A-Z, a-z veya 0-9 arasında ise 0 dan farklı, değilse 0 gönderir	isalnum('a')	1
isascii(c)	c bir ASCII karakter ise 0 dan farklı, değilse 0 gönderir	isascii('a')	1
isdigit(c)	c bir rakam ise 0 dan farklı, değilse 0 gönderir	isdigit('4')	2
islower(c)	c a-z arasında ise 0 dan farklı, değilse 0 gönderir	islower('P')	0
isupper(c)	c A-Z arasında ise 0 dan farklı, değilse 0 gönderir	isupper('P')	4
toascii(c)	c sayısı ile verilen ASCII koda sahip karakteri verir	toascii(65)	A
tolower(c)	c karakterini küçük harfe çevirir	tolower('D')	d
toupper(c)	c karakterini büyük harfe çevirir	toupper('b')	B

```c
/* ASCII kodaları 32-127 arasında olan karakterler üzerinde
 ctype.h kütüphanesinde tanımlı bazı makroların kullanımı */

#include <stdio.h>
#include <ctype.h>
main(void)
{
int  i;
char c;
for(i=32;i<127;i++)
{
c = toascii(i);
printf("%d\t%c  %c  %d  %d\n",i,c,tolower(c),isalpha(c),isdigit(c));
}
}
```

ÖDEV 6:

1. 0 dereceden 180 dereceye kadar tüm açıları cos, sin ve tan olarak düzgün bir tablo şeklinde sıralayan programın algoritma akış diyagramı ve programını yapınız.

Açı:	cos:	sin	tan
0	1.0000	0.000	0.000
1	0.9998	0.0175	0.0175

2. Sayısal loto programı yapılacaktır. Ancak aynı altılı gurubun içinde hiçbir şekilde aynı sayı ikinci defa gelmemelidir. Buna göre oluşturulacak programın algoritma akış diyagramı ve programını yapınız.

3. Yapacağınız program sizden isminizin girilmesini isteyecektir. Ancak isim girerken yanlışlıkla rakam girilirse program yanlış isim girdiğinizi söyleyerek sizi uyaracak ve ismi tekrar girmenizi isteyecektir.

UYGULAMA

```
C:\Documents and Settings\Administrator\Desktop\...
asagidaki tablodan islem turunu seciniz
1        Toplama
2        Cikartma
3        Carpma
4        Bolme
5        Karekok alma
6        Karesini alma
7        x^y alma
8        Sinus alma
9        Cosinus alma
10       Tanjant alma
11       Program sonlandir

--------------------------------
Process exited with return value 0
Press any key to continue . . . _
```

Yukarıdaki ekran görüntüsüne göre çalışan menü dönüşlü bir bilimsel hesap makinesi yapınız.