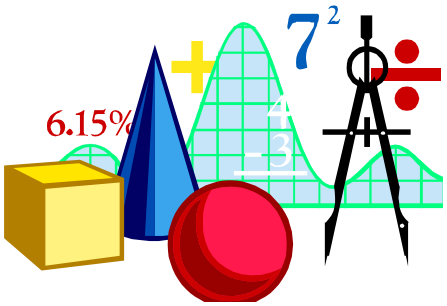
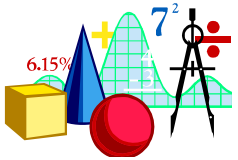
Karakter- és sztringkezelő függvények, matematikai függvények

- Karakterkezelő könyvtári függvények
- Mintaprogram a karakterosztályozásra
- Sztringek kezelése
- Sztringkezelő függvények
- Sztring argumentum átadása függvénynek
- Matematikai függvények
  - Szögfüggvények és inverz függvényeik
  - Exponenciális és logaritmikus függvények
  - Egyéb matematikai függvények
- Véletlenszám-generálás függvényei



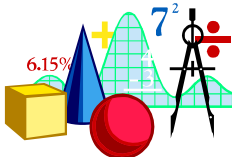
Karakterkezelő könyvtári függvények



A karakterkezelő függvényeket használó program elején meg kell adnunk a ctype.h header-fájlt :

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

A karaktereket kezelő makrók (a makrókat részletesen lásd később) formailag a függvények mintájára használhatók. Az alábbiakban mutatott makrók a paraméterként megadott karakterről eldöntik, hogy a karaktereknek egy bizonyos csoportjába tartozik-e.



Ha igen, akkor a makró nem nulla értéket ad, egyébként nullát.

isalnum(karakter)

igaz, ha a karakter alfanumerikus ('a'..'z', 'A'..'Z', '0'..'9').

isalfa(karakter)

igaz, ha a karakter betű : ('a'..'z', 'A'..'Z').

isdigit(karakter)

igaz, ha a karakter számjegy ('0'..'9').

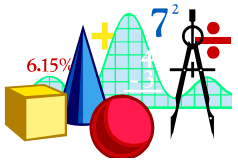
islower(karakter)

igaz, ha a karakter kisbetű ('a'..'z').

isupper(karakter)

igaz, ha a karakter nagybetű ('A'..'Z').

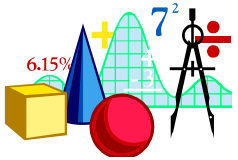
Mintaprogram a karakterosztályozásra:



A beadott karaktert megvizsgáljuk, mely karaktercsoportnak tagja. Egyes billentyűk, pl. a kurzormozgató és a funkcióbillentyűk két karaktert jelentenek: egy '\0' és egy nyomtatható karaktert.

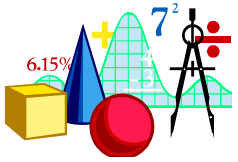
```
#include <ctype.h>
#include <stdio.h>
#define  ESC  27

main()
```




```
{
 char bill ;
 puts("Karakterosztalyozas" );
 do
  {
   printf("Nyomjon le egy billentyut:" );
   bill = getch(); putch(bill);
   if ( isalnum( bill ) )   puts("\n  alfanumerikus karakter" );
   else puts("\n  egyéb karakter (irasjel, \n    \
                                vezerlokarakter, vagy nem ASCII karakter)" );
   if ( isalpha( bill ) )   puts("  betu karakter");
   if ( isdigit( bill ) )  puts("  decimalis szamjegy karakter" );
   if ( islower( bill ) )   puts("  kisbetü" );
   if ( isupper( bill ) )   puts("  nagybetü" );
  }
 while ( bill  !=  ESC );
}
```

Sztringek kezelése

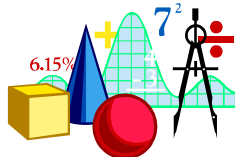


Emlékeztetőül: a sztring a C nyelvben nem önálló típus, hanem egy '\0' karakterrel záródó karaktervektort értünk alatta. A karaktervektor-jelleg miatt egy szövegváltozónak nem adhatunk egy értékadó utasítással értéket:

```
char szoveg[ 23 ];
szoveg = "ablak";          // hibás értékadási kísérlet
                           // mert szoveg egy mutatókonstans!
```

Az értékadás csak a karakterek egyenkénti átmásolásával, vagy ezt helyettünk elvégző függvénnyel lehetséges.

Sztringkezelő függvények:

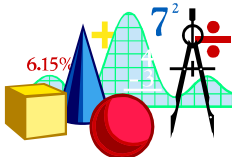


Sztring karaktereinek átmásolása egy másik sztringbe (értékadás sztringnek):

strcpy(<célsztringváltozó>, <forrássztring>);

Pl.:
```
char nev[15], name[20];
strcpy( nev, "Dennis Ritchie" );
strcpy( name, nev );
```

A másolás a szöveg végét jelző '\0' karaktert is átmásolja. Vegyük észre, hogy a függvény mindkét paramétere **char*** típusú mutató.



Szöveg hosszának meghatározása ('\0' nélkül):

<hossz> = **strlen**(<szöveg>);

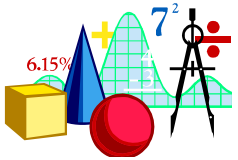
Pl.: meret = **strlen** (nev); // *meret 14 lesz*

Szöveg hozzáfűzése egy másik szöveg végéhez:

strcat(<szöveg>, <hozzáfűzött_szöveg>);

Pl.: **char** szov[20] = "tanulás";
strcat(szov, " szükséges");

Megj.: A <szöveg> vektornak elegendő hosszúnak kell lennie <hozzáfűzött_szöveg> befogadására.

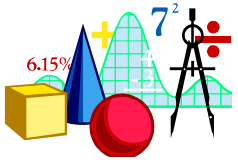


Keresett szöveg első előfordulásának megkeresése egy másik szövegben. A <karaktermutató> a megtalált előfordulás első karakterére mutat (vagy NULL, ha nem talált):

<karaktermutató> = strstr(<szöveg>, <keresett>);

Pl.:

```
char *poz, szov[12] = "zászlós",
keresett[3] = "ló";
poz = strstr( szov , keresett);
puts( poz );                    // kiírás: lós
```

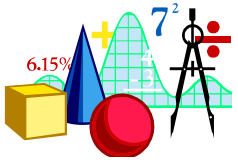


Szöveg adott számú első karakterének átmásolása egy célszövegbe, '\0' karakter elhelyezése nélkül:

strncpy(<célszöveg>, <forrásszöveg>, <hossz>);

Pl.:
```
char cel[ 12 ], forras[ 7 ] = "kapocs";
strncpy( cel, forras, 3 ); cel[ 3 ] = '\0' ;
puts( cel ) ;                                  // "kap"
```

Két szöveg összehasonlítása (komparálása) karakterről-karakterre, a karakterek kódjai alapján:



<viszonyszám> = strcmp(<szov1>, <szov2>);

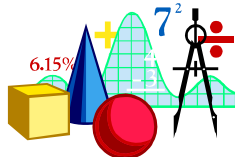
Ha szov1 < szov2, akkor viszonyszám < 0,
ha szov1 == szov2, akkor viszonyszám = 0,
ha szov1 > szov2, akkor viszonyszám > 0.

Pl.:
```
char s1[ 7 ] = "Góliát",  s2[ 6 ] = "Törpe";
egeszvalt = strcmp(s1, s2);   // egeszvalt < 0
```

A fenti szövegkezelő függvények használatához a string.h header-fájlt deklarálni kell a program elején:

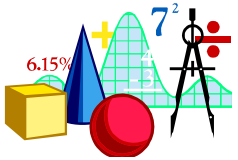
```
#include <string.h>
```

Sztring argumentum átadása függvénynek

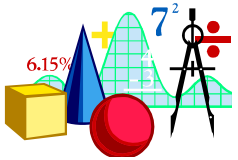


Sztringek átadása hasonló a vektorok átadásához, azzal a megjegyzéssel, hogy nem kell a vektor hasznos elemeinek számát külön argumentumként átadni, mert a '\0' karakter jelzi a karaktervektor utolsó hasznos elemét. Példaként írjunk függvényeket a Pascal nyelvből ismert **delete**, **insert** és **copy** alprogramok által megvalósított funkciókra és alkalmazzuk azokat programban!

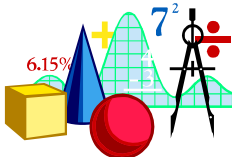
```
#include <stdio.h>
#include <string.h>          // strlen() miatt
void del(char mibol[ ], int honnan, int mennyit );
void insert( char mit[ ], char mibe[ ], int hova);
void copy(char *mibe, char *mibol, int honnan, int mennyit);
```




```
main( )
{
   char s1[ 11 ] = "siklas";
   char s2[ 20 ] ;
   printf( "%s\n", s1 );          //     siklas
   del(s1,1,3);
   printf( "%s\n", s1 );          //     sas
   insert( "zor",s1,1);
   printf( "%s\n", s1 );          //     szoras
   copy( s2,s1, 2,4 );
   printf( "%s\n", s2 );          //     órás
}
```

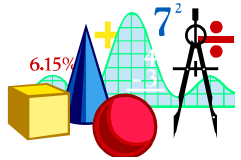



```
void del(char mibol[ ], int honnan, int mennyit )
{
 int i, regimeret ;
 regimeret = strlen( mibol );
 for ( i=0;  i < regimeret-honnan && i < mennyit;  i++)
  mibol[ honnan+i ] = mibol[ honnan+mennyit+i ];
 while (honnan+mennyit+i < regimeret+1)
  {mibol[ honnan+i ] = mibol[ honnan+mennyit+i]; i++;}
}
void insert( char mit[ ], char mibe[ ], int hova)
 {                                                  //  hova 0-tól indul
  int hosszt, hosszb, i;
  hosszt = strlen(mit);
  hosszb = strlen(mibe);
  for (i = 0; i< hosszb – hova + 1; i++) // helycsinálás, eltolás jobbra
     mibe[ hosszb + hosszt - i ] = mibe[ hosszb - i ];
  for ( i = 0;  i < hosszt;  i++ )
     mibe[hova+i ] = mit[ i ];                  // bemásolás
}
```




```
void copy(char *mibe, char *mibol, int honnan, int mennyit)
//vagy  void copy(char mibe[ ], char mibol[ ], int honnan, int mennyit)
{
  char * smut1, * smut2 ;                  // segéd mutatók
  int i ;
  smut1 = mibe ;
  smut2 = mibol + honnan ;
  for (i = 0; mibol[ honnan+i ] && i < mennyit;  i++, smut1++, smut2++)
    *smut1 = *smut2;                       // kimásolás
  *smut1 = '\0' ;
}
```

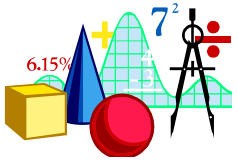
Matematikai függvények



Az alábbiakban néhány fontosabb matematikai függvényt ismerhetünk meg, melyek használata előtt a **math.h** include fájlt be kell emelni a programba.

A függvények hiba esetén az **errno** változó értékének beállításával jelzik a hiba fajtáját: ha a megadott argumentum nincs benne a függvény értelmezési tartományában, errno értéke **EDOM** lesz (33), ha az eredmény nem fér el a számára megadott típusban, errno értéke **ERANGE** lesz (34). A függvények általában double argumentumból double típusú eredményt állítanak elő (d $\Rightarrow$ d).

Szögfüggvények és inverz függvényeik:



sin(<szög_radiánban>)

a megadott szög szinuszát adja (d⇒d).

asin(<argumentum>)

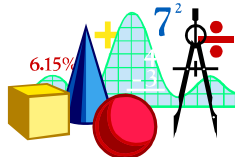
a -1..+1 itervallumba eső argumentum arkusz szinusz szögértékét adja radiánban (d⇒d). A zárt intervallumból kieső argumentum EDOM hibakódot ad. A szög a -π/2..+π/2 zárt intervallumban adódik.

cos(<szög_radiánban>)

a megadott szög koszinuszát adja (d⇒d).

acos(<argumentum>)

a -1..+1 itervallumba eső argumentum arkusz koszinusz szögértékét adja radiánban (d⇒d). A zárt intervallumból kieső argumentum EDOM hibakódot ad. A szög a 0..+π zárt intervallumban adódik.



tan(<szög_radiánban>)

a megadott szög tangensét adja (d⇒d). Ha a -π/2..+π/2 intervallumban megadott szög a határokhoz közel van, a függvény értéke 0 lesz és ERANGE hibakódot kapunk.

atan(<argumentum>)

az argumentum arkusz tangens szögértékét adja radiánban (d⇒d).

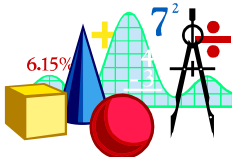
A szögérték a -π/2..+π/2 nyílt intervallumban adódik.

atan2(<y>,<x>)

az y/x hányados arkusz tangens szögértékét adja radiánban (d⇒d).

A függvény a teljes -π..+π intervallumra helyes szögértéket ad. Ha x==y==0, EDOM hibakódot ad.

Exponenciális és logaritmikus függvények



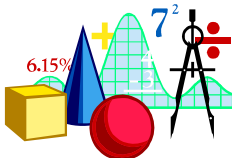
exp(<argumentum>)

a természetes alapú logaritmus **e** alapszámát emeli az argumentumnak megfelelő hatványra (d⇒d). A nem ábrázolható nagyságú, vagy kicsiségű eredmény esetén ERANGE hibakódot állít be.

log(<argumentum>)
log10(<argumentum>)

az argumentum természetes alapú, ill. tízes alapú logaritmusát adja (d⇒d). Nempozitív argumentum EDOM hibakódot eredményez.

Egyéb matematikai függvények

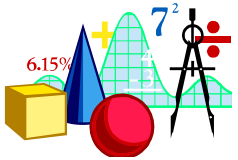



```
abs(<argumentum>)      // (int⇒int)
labs(<argumentum>)     // (long⇒long)
fabs(<argumentum>)     // (d⇒d)
```

függvények az adott típusú argumentum ugyanolyan típusú abszolutértékét adják.

```
floor(<argumentum>)
```

az argumentumtól kisebb legnagyobb egész számot adja (d⇒d), pl. 5.25⇒5; -7.2⇒ -8. Kerekítés esetén floor(argumentum + 0.5) írandó.



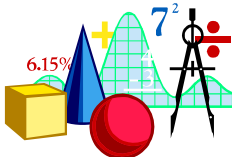
pow(<x>,<y>)

a x-nek az y kitevőre emelt hatványát adja (d⇒d). Túlcsordulást ERANGE hibakóddal jelzi. Ha x negatív és y nem egész, EDOM hibakódot ad.

sqrt(<argumentum>)

az argumentum négyzetgyökét adja (d⇒d). Negatív argumentum EDOM hibakódot ad.

Véletlenszám-generálás függvényei



srand(time(NULL));

függvény alapállapotba hozza a véletlenszám-generátort.
Használatához a time.h fájl beemelése szükséges.

rand() % max

0..(max-1) közötti véletlenszámot ad.