Függvények

- *Függvények deklarációja és prototípusa*
- *A függvénydefiníció*
- *Hivatkozás függvényre*
- *Mintaprogram függvénydefiniálással*
- *Címszerinti argumentumátadás*
- *Vektorok átadása függvénynek*

Függvények deklarációja és prototípusa

A C nyelvben a függvény központi fogalom. Egy függvénynek a main()*-nek minden programban egyszer és csak egyszer szerepelnie kell. Egy függvény egy általában többször kiszámítandó értéket határoz meg, vagy logikailag egységet képező feladatot végez el. A fejlesztőrendszer részét képező könyvtári függvényeket, vagy az általunk készítetteket egyaránt használhatjuk. A függvényeket felhasználás (hivatkozás) előtt deklarálni kell, és saját függvényeinket egy helyen a programban teljesen meg kell adni, azaz definiálni szükséges. A prototípus a deklarációtól abban több, hogy megadja a függvény paramétereit is:*

<típus> függvénynév (<paraméterek>) ;

Pl.: **int** *minimum*(**int** *a*, **int** *b*) ;

A <típus> határozza meg a függvény által visszaadott érték típusát (pl. **int**, **float**).
Ez összetett, **struct** *és* **union** *típus is lehet, valamint* **void***, amely értéket vissza nem adó függvényt (eljárást) eredményez. A függvény visszatérési értékét a függvény definíciós blokkjában elhelyezett* **return** *utasítás után álló kifejezés értéke adja. A* **return void** *függvényeknél elmaradhat.*

A <paraméterek> *megadásánál ügyelni kell arra, hogy minden paraméter előtt meg kell adni a típusát, amely az eddig tanultak bármelyike lehet. A prototípust a végén álló* **;** *különbözteti meg a függvénydefiníció fejlécétől. A prototípusok a program, vagy a blokkok deklarációs részében helyezhetők el.*

- **A függvénydefiníció** *a prototípusra emlékeztető fejlécből és egy blokkból áll:*

```
<típus> függvénynév (<paraméterek>)
{
  <deklarációk>     // lokálisak, blokkon belül érvényesek
  <utasítások>      // köztük lehet a return utasítás
}
```

A paraméterek csak érték szerinti átadásúak.
A cím szerinti átadást mutatóval kell megvalósítani.
A függvénydefiníciók általában a program végén helyezkednek el, a prototípusok és a hivatkozások után. ***Amennyiben a definíció megelőzi a hivatkozásokat, akkor a prototípus elmaradhat****. Függvénydefiníció nem adható meg más függvény blokkjában, csak deklaráció. A függvénydefiníciók egymás mellé rendeltek, a* main() *függvénnyel azonos szinten, moduláris módon.*

Példa egy függvénydefiniálásra:

```
int minimum( int a, int b)
{
    return (a < b) ? a : b ;
}
```

Hivatkozás függvényre (függvények használata)

A C nyelv nem void függvényei pontosvessző nélkül függvénykifejezésként kifejezésben alkalmazhatók, de sok esetben pontosvesszővel lezárva utasításként kerülnek felhasználásra.

A függvényhívás formája:

függvénycímkifejezés (<argumentumok>)

A függvénycímkifejezés lehet a konstans mutatóként viselkedő függvénynév, vagy más, a függvény címét szolgáltató kifejezés. Az <argumentumok> vesszővel elválasztott kifejezések, melyek számban és (esetleges automatikus konvertálás után) típusban egyeznek a prototípus, vagy a definíció paramétereivel. Amennyiben a paraméterlista helyén a void (=üres) szó szerepelt, nem adható meg argumentum.
Az érték szerinti átadás miatt a függvény nem tud visszaadni megváltoztatott értéket az argumentumváltozóknak.

Példa függvényhivatkozásra:

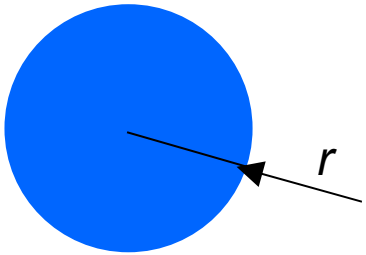
```
printf("k és m közül a kisebb értéke= %d", minimum( k , m ) );
```

- ***Mintaprogram függvénydefiniálással:*** hatványozás

```
#include <stdio.h>
#include <math.h>          // pow( ), exp( ), log( ) miatt
double hatvany(double x, double y);   // prototípus
void main()
{
    double alap, kitevo;
    printf("Hatványalap = ");  scanf( "%lf", &alap );
    printf("Hatványkitevö = ");  scanf( "%lf", &kitevo );
    printf("\nEredmény= %lf", hatvany(alap,kitevo) );
    printf("\nPow-rel= %lf\n", pow(alap,kitevo) );
    getch();
}
double hatvany(double x, double y)      // definíció
{
    return exp(y*log(x));
}
```

Címszerinti argumentumátadás

Írjunk függvényt a kör területének és kerületének számítására a sugár ismeretében.



A terület és kerület argumentumváltozók másolata jön létre a függvénybe való belépéskor, melyek megváltozott értéküket a függvényből való kilépéskor elveszítik, változatlanul hagyva eközben a terület és kerület változókat. Megoldás: a változók címét adjuk át a függvénynek, melyet, mint mutatóértéket használva módosítja a mutatott terület és kerület változókat.

Címszerinti argumentumátadás ..

```
#include <stdio.h>
void kor ( float r, float* ter ,float* ker );     // prototípus
main()
{
  float sugar, t, k;
  printf( "Sugar =" );
  scanf( "%f", &sugar );
  kor(sugar, &t, &k);                             // hivatkozás
  printf("Terulet= %10.2f, kerulet= %10.2f ",  t, k);
}
void kor( float r, float* ter ,float* ker )      // definíció
{
  float Pi = 3.1415;
  *ter = r * r * Pi;
  *ker = 2 * r * Pi;
}
```

Vektorok átadása függvénynek

A vektorelem típusának megfelelő típusú mutató értékét kell átadni, mely a vektor első elemére mutat, majd pointeraritmetikával lépegetni a vektor elemein. Hogy ki ne lépjünk a vektorból, át kell adni a függvénynek a vektor elemeinek a számát is, vagy észlelni kell az elem értékéből, hogy az az utolsó, mint ahogy mutatja ezt a szövegtárolásra használt karaktervektorok '\0' értékű utolsó eleme.

Pl.:

```
float vektor[ 7 ] , atlag;
int elemszam = 7;
float szum( float * vekt, int db)     // vagy float vekt[ ]
{
    float sv = 0;
    int i = 0;
    for ( ; i < db; i++)
    sv += *(vekt + i );                    // vagy vekt[ i ];
    return sv;
}
atlag = szum(vektor, elemszam) / elemszam; // Hivatkozás:
```