Függvények

Felhasználási előnyök, területek:
- modularizálás
- kódismétlések elkerülése

Függvény deklarációja: Amikor bejelentjük egy függvény létezését.

(*visszatérési érték típusa*) (*függvény neve*) (*argumentumlista*);

Egy program tetszőleges mennyiségben tartalmazhatja egy függvénynek a deklarációját, azaz akárhányszor lehet kijelenteni a létezését.

A C programozási nyelvben a függvény neve határozza meg a függvényt magát, a visszatérési érték típusa és az argumentumlista nem.

Következmények:
- nem lehet ugyanazzal a névvel, de különböző visszatérési érték típussal és/vagy különböző argumentumlistával függvényeket deklarálni
- a C nyelv ***nem*** támogatja a *generikus programozást*, azaz a típusfüggetlen megoldásokat (függvények, struktúrák)

Feladatok:
- Próbáld ki - fordítva és futtatva - egy egyszerű kódban (egy *main* függvényt tartalmazó forrásfile-ban) az alábbiakat:
  - deklarálj egy függvényt, majd másold le ugyanezt a következő sorba
  - deklarálj egy függvényt, majd ugyanezzel a névvel, de más visszatérési érték típussal és/vagy argumentumlistával legyen egy másik függvényed deklarálva

Függvény definíciója: Megadjuk a függvényben, hogy milyen utasításokat hajtson végre.

(*visszatérési érték típusa*) (*függvény neve*) (*argumentumlista*) { *függvénytörzs* }

A függvény deklarációja elhagyható, ha van definíció.
A függvény deklarációja és definíciója meg kell, hogy egyezzen.

Feladatok:
- Próbáljuk ki hasonlóan az előző feladatokhoz, hogy deklarálunk egy függvényt, majd
  - ugyanezzel a szintaktikával definiáljuk is (a törzsbe tetszőleges utasításokat adjunk)

Függvény hívása:

(*függvény neve*) (*argumentumok*);

A hívó félnek nem kell tárolnia a hívott függvény visszatérő értékét.

Visszatérési érték típusa:
- A függvény típusa a visszatérési érték típusa.
- Tetszőleges, a típusok bevezetőben található built-in vagy user-defined típussal is visszatérhet egy függvény. A ***void*** esetben nincs visszatérési érték. A hívó félnek a ***return***

utasítással adható át érték, akár egy lokális vagy globális változó, akár egy határozott érték (ilyenkor implicit casting gyakran van).

- Egy függvényt úgy is deklarálhatunk, definiálhatunk, hogy nem adjuk meg a visszatérési érték típusát. Ekkor default *int* lesz a visszatérési érték típusa (lásd helloworld bevezető). Erősen nem javasolt a használata! Erre a fordító is figyelmeztet. Szintén fordító függő, hogy ilyenkor a default 0 értékkel tér vissza vagy nem. Továbbra is erősen nem javasolt a függvények ilyen deklarációja és definíciója!
- A *void* függvényeket leszámítva minden függvényben szükséges legalább egy *return* utasítás a függvény hatáskörében értelmezve. Több is lehet, *if..else* elágazásokon belül, sőt, ez egy közkedvelt eljárás a függvény futásának megszakítására.

Feladatok:

- Definiáljunk egy függvényt, tetszőleges névvel és argumentumlistával, visszatérési érték típusának megadása nélkül és üres függvénytörzzsel. A *main* függvényben hívjuk meg és irassuk ki az értékét *printf(“%d”, …);* segítségével.
- Adjunk *void*-t a visszatérési érték típusának a fenti függvényben és így próbáljuk meg fordítani a kódot.
- Módosítsuk ugyanezt a függvényt lebegőpontos visszatérési típusra.
- Írjunk egy függvényt, aminek *float* a visszatérési érték típusa és nincs a definíciójában *return* utasítás.
- Írjunk függvényt *int* visszatérési érték típussal, amiben egy *if* ágban van *return*, amennyiben az argumentumként kapott szám negatív. Ekkor térjen vissza a függvény nullával.

Argumentumlista:

- Tetszőleges, a típusok bevezetőben található built-in vagy user-defined típus is átadható egy függvénynek.
- Tetszőleges mennyiségű argumentum szerepelhet, vesszővel elválasztva.
- Amennyiben üres a függvény argumentumlistája, úgy a függvény hívásakor argumentumokat is írhatunk üres zárójelek helyett. Ezek az átadott értékek azonban nem elérhetőek a függvényen belül (valójában elérhetőek, de az már nem C kóddal, hanem assembly kóddal, registereken keresztül). Ha el akarjuk kerülni az ilyen hívási lehetőséget, akkor az argumentumlista helyére *void*-t írva fordítási hibát kapunk.
- Az átadott argumentumok nem a hívó fél változói, hanem lokális változók lesznek, új memóriaterületen tárolva. Azaz, bármilyen változtatást is hajtunk végre rajtuk, az nincs hatással a hívó részen lévő változókon.
- Kivétel ez alól, ha tömböt adunk át. Egy tömb átadása függvénynek a következő módon tehető meg:

  return_type function(int size, array_type array[size]);

  A tömb méretét is át kell adnunk, amennyiben a tömb méretének megadásához használni szeretnénk. A tömb méretét az argumentumlistában elhagyhatjuk, sőt, az első argumentumként használt *size*-t is: ekkor azonban a függvényben ismeretlen a tömb mérete, hivatkozni se tudunk rá és így potenciálisan kifuthatunk a programunk memóriakeretén túlra.
  Tehát erősen javasolt a fenti szintaxis használata!

  Tömb argumentum esetében a függvényben módosított értékek a hívó fél oldalán is megtörténnek. Ennek egyszerű oka van: a tömb nem másolódik át, csupán a tömb memóriacímét tartalmazó (első helyiértékre hivatkozó) mutató (ez is növeli a C hatékonyságát). Tehát valójában ez a függvény kinézete:

  *return_type function(int size, array_type * array);*

  Bővebben a pointerek részen.

Feladatok:

- Készítsünk egy függvényt, üres argumentumlistával. Hívjuk meg:
  - üresen, pl. *foo()*
  - érték(ek)kel mint argumentummal, pl. *foo(1, 2, 3)*
- Módosítsuk a függvényt void argumentumlistára, pl. *void foo(void),* és így próbáljuk meghívni nem-üres argumentummal.
- Legyen egy függvényünk, tetszőleges visszatérési érték típussal, argumentumlistaként a tömb méretével és egy tetszőleges típusú tömbbel, pl *void foo(int c, int a[c]);*
  A *main*-ben hívjuk meg egy ott definiált tömb változóval és irassuk ki printf-fel a tömb első értékét a függvény hívása után:
  - töröljük a függvény deklarációjánál, a tömb argumentum megadásánál a méretet, nézzük meg, hogy fordul a kód
  - töröljük a függvény deklarációjánál az első argumentumot (méretet megadó argumentum)
  - definiáljuk a függvényt, amiben a tömb nulladik elemét módosítjuk, fordítsuk le és futtassuk: a main-ben kiírt tömbelem változott a függvényben módosítottra
  - módosítsuk a függvény argumentumlistájában a tömböt, tegyünk const kulcsszót a tömb elé, pl. void *foo(int c, const int array[s]);*
    A fordító hibával tér vissza: a függvényben nem módosítható az argumentumként átadott tömbelem.

Rekurzív függvények: Olyan függvény, ami önmagát hívja.

(*visszatérési érték típusa*) (*függvény neve*) (*argumentumlista*)
{
    utasítások1;
    függvény önhívása;
    utasítások2;
}

Akár az első, akár a második utasításcsoport elhagyható, sőt, mindkettő is. A legegyszerűbb rekurzív függvény: *void foo() { foo(); }.*

Figyelem! A rekurzív függvény úgy működik, hogy a memóriába új példányt helyez, memóriafoglalással, új lokális változókkal, új utasításokkal. Az elsővel van a probléma: véges memóriánk van! Ez segmentation fault-hoz vezet általában.

Sőt, nem is hatékony a rekurzió, mivel a memóriában nagyobb távolságokat kell megtenni a program futásakor. Azaz, ugyan matematikailag nagyon szép kódokat eredményez a rekurzió használata, de potenciális memóriakezelési problémákat okoz és lassíthatja a program futását is.

Egy rekurzív függvény megvalósítható ciklussal is, általában javasolt is inkább azok használata. Van eset, amikor a fordítók át tudják alakítani a rekurzív függvényünket ciklussá.
Ez a *tail recursion (*farokrekurzió) esetében mindig sikerül is. Ekkor a függvény önhívását már nem követi további utasítás, pl a fenti legyegyszerűbb példa is.

Végtelen ciklus rekurzióval: ha nincs megszakítási (azaz, olyan eset, amikor már nem hívjuk meg a függvényt ismételten) feltétel megadva.

Tehát a rekurzív függvény implementálásakor első és legfontosabb feladatunk a kilépési feltétel megadása, pl.

```
int foo(int n)
{
        if(n > 10)
                return 0;
        return foo(n + 1);
}
```

Feladatok:

1. A *main*-ben legyen egy 10 elemű int tömbünk, tetszőleges int értékekkel feltöltve:
   - Írjunk függvényt, ami megkeresi a minimumát és visszatér ezzel az értékkel.
   - Írjunk függvényt, ami megkeresi a maximumát és visszatér ezzel az értékkel.
   - Egyesítsük a két függvényt eggyé! Tipp: használjunk egy struct-t, amiben letároljuk az értékeket és ezzel térjünk vissza.
2. Fibonacci számok kiszámolása rekurzívan: Kérjünk be egy számot a felhasználótól, majd írjuk ki az adott számhoz tartozó Fibonacci számot. Használjunk rekurzív függvényt ehhez!
3. Deklaráljunk egy *struct*-t, ami tárolja személyek adatait, úgymint age, height, id, mindegyik int típusú. Legyen ebből egy 5 elemű tömbünk a *main*-ben, feltöltve tetszőleges, típushelyes adatokkal.
   - Írjunk függvényt, aminek átadjuk ezt a tömböt és kiszámoljuk az átlag magasságot, amivel visszatér a függvény.
   - Írjunk egy rekurzív függvényt, ami a bemenetként ezt a tömböt kapja, és kiszámolja a személyek átlag életkorát.
4. Bónusz: Függvények írása, melyben az összeadást, szorzást bitműveletekkel oldjuk meg. (https://stackoverflow.com/questions/3722004/how-to-perform-multiplication-using-bitwise-operators)