7. óra

# Függvények

## Mire jó?

Szebb lesz tőle a kód.

Könnyebb lesz a debugolás.

Nem kell többször leírnunk ugyanazt az eljárást.

## Deklaráció

típus függvénynév(paraméterek, listája){

függvény törzse  
 visszatérési érték (return)  
}

int main(){  
 Ide jön a kód.  
 return 0;  
}

## Függvényhasználat a gyakorlaton

Mostantól típusonként egy-egy beolvasó függvény lesz. Beolvasás(), tartalma: N=egészet\_beolvas(); X=tömböt\_beolvas(); stb.

int egészet\_beolvas(){  
 …;  
 return(…);  
}

void Beolvasás(){  
 N=egészet\_beolvas();  
 //nincs return.  
}

Mostantól a fő függvény

## Paraméterek, értékátadás

Cím szerint vagy érték szerint lehet átadni. (Írásvédett, vagy módosítható.)

Deklaráció:  
float Paraméteres\_függvény\_tömbbel(const int a, float &b, string[] szöveg\_tömb){…} //"&" jel a cím szerinti átadás, a memóriában mutató (pointer), amin keresztül elérjük (és módosítjuk!) a b-t.  
//const egy olvashatósági flag, ha be van jelölve, akkor nem változtatható.

Meghívása:  
int i1=3; float f1=.0; string str[2]={"Hi","Bye"};  
float F = Paraméteres\_függvény\_tömbbel(i1, f1, \*str); //paraméter neve nem kell, hogy megegyezzen! Lehetne szöveg\_tömb is, de az str rövidebb.

Lefut a Paraméteres\_függvény\_tömbbel, felszabadul minden paraméter a memóriából, DE a cím szerint átküldött paraméterek módosultak. (Természetesen a visszatérési érték (return) is megmarad az F-ben.)

## Kitérő: Összetett programozási tételek és a másolás

tömb → tömb  
tömb → tömbök

másolás: elemi. Igaz, hogy tömbhöz tömböt rendel, de egyesével teszi, ezért elemi, nem összetett.

## Beadandóhoz segítség

Kétdimenziós tömb, táblázat: N×M (N: helyek, M: időpontok)

float Adat[N][M]

int A[1][2];

A[0][1]=1;

cout<<A[0][1];

## HF

8. Írjon programot, amely kiszámítja két vektor skaláris szorzatát!