Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedés- és Járműirányítási Tanszék

Programozás C- és Matlab nyelven
C programozás kurzus
BMEKOKAM603
Függvények

Dr. Bécsi Tamás

6. Előadás

4. Függvények
Bevezetés

- Egy idő után az egyetlen `main(){}` függvénnyel megírt programunk túl nagy méretű lesz.
- Vannak programrészek, amelyekre több helyen is szükségünk lehet.
- A túl bonyolult algoritmusok a teljes program átláthatóságát csökkentik.
- A fentiek miatt célszerű a program tagolása, ahol az egyes alfeladatokat el tudjuk különíteni, így strukturálva a programot.

4. Függvények

Forma:

```
visszatérési-típus függvénynév(argumentum deklarációk)
{
    deklarációk és utasítások
}
```

4. Függvények


```
//függvény deklaráció (prototípus)
void szamkiir(int szam);
//A main fv
int main(void) {
    szamkiir(10);
    return 0;
}
//függvény definíció
void szamkiir(int szam){
    printf("%d",szam);
}
```

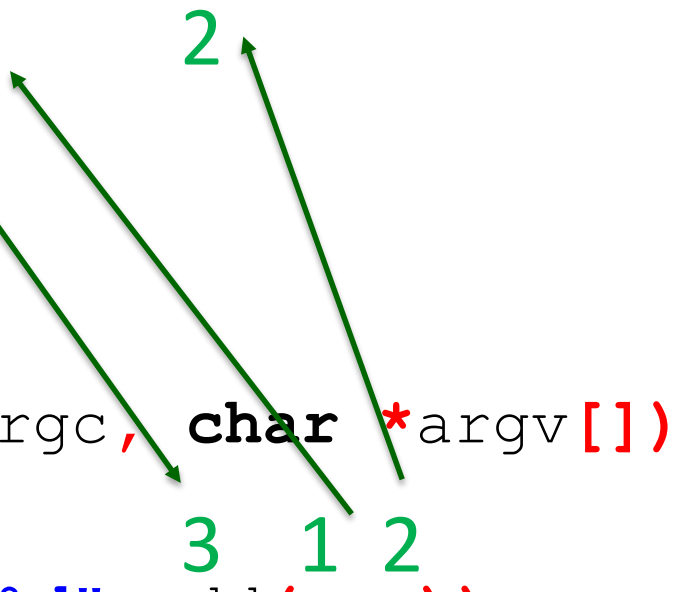
4. Függvények visszatérés

- A hívott függvény a hívó függvénynek a return utasítással adhat vissza értéket, melyet tetszőleges kifejezés követhet:
- Formája: `return kifejezés;`
- Ha szükséges, a kifejezés típusa a visszatérési típusra konvertálódik.
- A return utáni kifejezés opcionális. Adott esetben a return is elhagyható.
- A return utasítás a meghívásának pontján tér vissza a hívott függvényből, akkor is ha utána még más kódrészletek találhatóak.

4. Függvények példa


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

int main(int argc, char *argv[])
{  int x=1,y=2;
   printf("x+y=%d",add(x,y));
   return 0;
}
```



4. Függvények
return példa (Szignumfüggvény)


```
int signum(int szam){
 if (szam>0) return +1;
 if (szam<0) return -1;
 return 0;
}
int main(void) {
 printf("%2d\n",signum(10));
 printf("%2d\n",signum(-10));
 printf("%2d\n",signum(0));
   return 0;
}
```

$$\operatorname{sgn} x = \begin{cases} -1, & \text{ha } x < 0 \\ 0, & \text{ha } x = 0 \\ 1, & \text{ha } x > 0 \end{cases}$$

```
 1
-1
 0
--------------------------------
Process exited with return
value 0
Press any key to continue . . .
```

4. Függvények
Paraméterátadás

- A függvények a paramétereket érték szerint veszik át, azaz amennyiben a paraméter értéke a függvényen belül változik, ez nincs hatással az átadott változó értékére.

4. Függvények
Érték szerinti paraméterátadás példa


```
void swap(int a, int b)
{
   int c;
   c=a;a=b;b=c;
   return;
}
int main(int argc, char *argv[])
{
 int x=1,y=2;
 swap(x,y);
 printf("x:%d y:%d",x,y);
 return 0;
}
```

1 2

1 2

a=2, b=1, de csak a függvényen „belül”

„x:1 y:2” Tehát nem cserélődött fel
A két változó értéke

4. Függvények
Paraméterátadás (tömb típus)

Az érték szerinti paraméterátadás „máshogy működőnek tűnik” tömb típusok esetén, erre magyarázatot a következő előadás ad.

„hamu”

```
void confuse(char s[]) {
    s[0]='m'; return;
}
int main(int argc, char *argv[]) {
    char s[]="hamu";
    printf("%s\n",s);
    confuse(s);
    printf("%s\n",s);
    return;
}
```

„mamu”

„hamu”

„mamu”

4.4. Az érvényességi tartomány szabályai

- Egy név érvényességi tartománya az a programrész, amiben a nevet definiálták.
- **Lokális változók:** Egy függvényen belül definiált változók. A helyi változók neve, értéke más függvényekben ismeretlen. Ide tartoznak a függvények argumentumai is.
- **Külső változók:** (globális változók) A függvényeken kívül definiált változók, hatókörük a deklarációjuk végén kezdődik, és az aktuálisan fordított fájl végéig tart. (beleértve a függvényeket.)

4.3. A külső változók példa


```
int kulso=0; //Külső változó deklarációja

void inckulso(){
    kulso++;
    return;
}
int main(void) {
    int i;
    for (i=1;i<5;i++)
    {
        inckulso();
        printf("kulso: %d\n",kulso);
    }
    return 0;
}
```

```
kulso: 1
kulso: 2
kulso: 3
kulso: 4

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

4.6. Statikus változók

- A belső változók a függvény minden egyes meghívásakor újra létrejönnek, és értéket kapnak.
- A statikus változók ezzel szemben a függvényre nézve lokálisak, de állandóan megtartják az értéküket két függvényhívás között is.

4.6. Statikus változók példa


```
int counter(){
   static int i=0;
   return ++i;
}
int main()) {
   int i;
   for (i=5;i>0;i--)
   printf("main: %d counter: %d\n",
   i,counter());
   return 0;
}
```

```
main: 5 counter: 1
main: 4 counter: 2
main: 3 counter: 3
main: 2 counter: 4
main: 1 counter: 5


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

4. Függvények
Rekurzív hívás (megemlítés)


```
int fakt (int i)
{
    return (i<2)?1:i*fakt(i-1);
}
```