Programozás I.

4. előadás
Összetett programozási tételek
Érték- és referencia típusú változók C#-ban
Metódusok C#-ban

Sergyán Szabolcs
sergyan.szabolcs@nik.uni-obuda.hu

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertechnológia Intézet

2012. október 1.

Tartalom

Tartalom

Tartalom

Másolás

Típusfeladatok

1. Egy osztály tanulóinak átlageredménye alapján határozzuk meg, hogy bizonyítványukba *jeles*, *jó*, *közepes* vagy *elégséges* kerül-e. (Tegyük fel, hogy bukott tanuló nincs.)
2. Egy szöveg minden magánhangzóját cseréljük ki az *e* betűre.

Közös jellemzők

- Az eredmény ugyanannyi elemszámú mint a bemenet
- Az eredmény i. elemét a bemenet i. eleméből lehet meghatározni

Másolás

Bemenet

X: Feldolgozandó tömb
N: Tömb elemeinek száma

Kimenet

Y: Eredmény tömb

Pszeudokód

Eljárás Másolás(X, N, Y)
  Ciklus $i \leftarrow$ 1-**től** N-**ig**
    $Y[i] \leftarrow$ művelet $X[i]$
  Ciklus vége
Eljárás vége

Másolás

Megjegyzések

- Az eredmény mindig ugyanannyi elemszámú mint a bemenet
- A *művelet* segítségével az egyszerű másoláson túl az egyes elemekkel egy-egy elemi műveletet is el lehet végezni (pl. másoljuk át a számok abszolutértékeit)
- Nem lehet az elemek közötti összefüggést kihasználni

Másolás

Magánhangzók *e*-re cserélése

Bemenet: X – karakter tömb
N – X elemeinek száma
Kimenet: Y – átalakított karakter tömb

Eljárás Másolás (X, N, Y)
  Ciklus $i \leftarrow$ 1-**től** N-**ig**
    Ha magánhangzó($X[i]$) **akkor**
      $Y[i] \leftarrow$ 'e'
    különben
      $Y[i] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Kiválogatás

Típusfeladatok

1. Egy személyzeti nyilvántartásban emberek neve és személyi száma szerepel, adjuk meg a 20 évnél fiatalabb lányokat.
2. Adjuk meg egy természetes szám összes osztóját.
3. Adjuk meg egy osztály azon tanulóit, akik jeles átlagúak.

Közös jellemzők

- Hasonlítanak a feladatok a keresésre, mert adott tulajdonságú elem(ek)et kell megadni, de nem csak egyet.
- Hasonlítanak a feladatok a megszámlálásra is, de nem megszámolni kell az elemeket, hanem megadni/másolni.

Kiválogatás

Bemenet

X: Feldolgozandó tömb
N: Tömb elemeinek száma
T: Tulajdonság függvény

Kimenet

Y: Eredmény tömb
DB: Y tömbbe kiválogatott elemek száma

Pszeudokód

Eljárás Kiválogatás (X, N, T, Y, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha $T(X[i])$ **akkor**
      $DB \leftarrow DB + 1$
      $Y[DB] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Kiválogatás

Megjegyzések

- Az eredmény tömb elemszámát nem lehet előre pontosan meghatározni, de biztos, hogy nincs több eleme mint a bemeneti tömbnek
- $X[i]$ helyett néha csak i-t másoljuk Y-ba, azaz a feltételnek megfelelő elemek indexét tároljuk

Kiválogatás

- Ha a bemeneti sorozatra már nincs szükség a későbbiekben, akkor a kiválogatás eredménye e bemeneti sorozat elejére is kerülhet
- Ebben az esetben kevesebb memóriát kell lefoglalnunk

Pszeudokód

Eljárás Kiválogatás(X, N, T, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha $T(X[i])$ **akkor**
      $DB \leftarrow DB + 1$
      $X[DB] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Kiválogatás

- Lehetséges megvalósítás az is, ha a kihagyandó elemeket megjelöljük valamilyen módon
- Ebben az esetben is elveszítjük az eredeti sorozatot

Pszeudokód

Eljárás Kiválogatás(X, N, T)
  Ciklus $i \leftarrow$ 1-**től** N-**ig**
    Ha $\neg(T(X[i]))$ **akkor**
      $X[i] \leftarrow$ speciális érték
    Elágazás vége
  Ciklus vége
Eljárás vége

Kiválogatás

Jeles átlagú tanulók neve

Bemenet: X – tanulók tömbje
N – X elemeinek száma
Kimenet: NEV – jeles tanulók neveit tartalmazó tömb
DB – NEV-be kerülő elemek száma

Eljárás Kiválogatás(X, N, NEV, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha $X[i].atlag \geq 4.71$ **akkor**
      $DB \leftarrow DB + 1$
      $NEV[DB] \leftarrow X[i].nev$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Szétválogatás

Típusfeladatok

1. Adott N darab különböző természetes szám, válogassuk szét a párosakat és a páratlanokat.
2. Az osztály tanulóit névsoruk alapján válogassuk szét lányokra és fiúkra.
3. Az osztály tanulóinak félévi átlageredményei alapján válogassuk szét jelesekre, jókra, közepesekre, elégségesekre, valamint elégtelenekre.

Közös jellemzők

- Egy sorozathoz több sorozatot rendelünk
- Lényegében egymás után kettő (vagy több) kiválogatással megoldhatók a feladatok
- Két kiválogatással megvalósítani viszont felesleges, mert amely elemeket nem választjuk ki a kiválogatásnál az egyik csoportba, azok tartoznak a másik csoportba

Szétválogatás

Bemenet

X:	Feldolgozandó tömb
N:	Tömb elemeinek száma
T:	Tulajdonság függvény

Kimenet

Y:	Egyik eredmény tömb
DBY:	X T tulajdonságú elemek száma
Z:	Másik eredmény tömb
DBZ	X $\neg T$ tulajdonságú elemek száma

Pszeudokód

Eljárás Szétválogatás(X, N, T, Y, DBY, Z, DBZ)
  $DBY \leftarrow 0$
  $DBZ \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha $T(X[i])$ **akkor**
      $DBY \leftarrow DBY + 1$
      $Y[DBY] \leftarrow X[i]$
    különben
      $DBZ \leftarrow DBZ + 1$
      $Z[DBZ] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Szétválogatás

Megjegyzések

- Y és Z elemszáma nem határozható meg előre, így N elemű tömböknek foglalunk helyet a memóriában

Szétválogatás

- Az eredmények egyetlen tömbbe is helyezhetők úgy, hogy a T tulajdonságúak az elején, a többi pedig a végén lesznek

Pszeudokód

Eljárás Szétválogatás(X, N, T, Y, DBY)
  $DBY \leftarrow 0$
  $INDZ \leftarrow N + 1$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha $T(X[i])$ **akkor**
      $DBY \leftarrow DBY + 1$
      $Y[DBY] \leftarrow X[i]$
    különben
      $INDZ \leftarrow INDZ - 1$
      $Y[INDZ] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Szétválogatás

Pszeudokód

Eljárás Szétválogatás(X, N, T, DB)
  $E \leftarrow 1$; $U \leftarrow N$; *seged* $\leftarrow X[E]$
  Ciklus amíg $E < U$
    Ciklus amíg $E < U$ **és** $\neg(T(X[U]))$
      $U \leftarrow U - 1$
    Ciklus vége
    Ha $E < U$ **akkor**
      $X[E] \leftarrow X[U]$; $E \leftarrow E + 1$;
      Ciklus amíg $E < U$ **és** $T(X[E])$
        $E \leftarrow E + 1$
      Ciklus vége
      Ha $E < U$ **akkor**
        $X[U] \leftarrow X[E]$; $U \leftarrow U - 1$
      Elágazás vége
    Elágazás vége
  Ciklus vége
  $X[E] \leftarrow$ *seged*
  Ha $T(X[E])$ **akkor**
    $DB \leftarrow E$
  különben
    $DB \leftarrow E - 1$
  Elágazás vége
Eljárás vége

Az elemek helyben, azaz a bemeneti tömbben történő szétválogatása is megoldható

Szétválogatás

Páros-páratlan szétválogatás

Bemenet: X – egészek tömbje
N – X elemeinek száma
Kimenet: Y – szétválogatott elemek tömbje
DB – páros számok száma

Eljárás Szétválogatás(X, N, Y, DB)
  $DB \leftarrow 0$; $IND \leftarrow N + 1$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    Ha páros(X[i]) **akkor**
      $DB \leftarrow DB + 1$; $Y[DB] \leftarrow X[i]$
    különben
      $IND \leftarrow IND - 1$; $Y[IND] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Metszet

Típusfeladatok

1. Adjuk meg két természetes szám osztóinak ismeretében az összes közös osztójukat.
2. Nyáron és télen is végeztünk madármegfigyeléseket a Balatonon. Ismerjük, hogy nyáron, illetve télen mely madárfajok fordultak elő. Állapítsuk meg ezek alapján, hogy melyek a nem költöző madarak.
3. Négy ember heti szabad estéi ismeretében állapítsuk meg, hogy a héten melyik este mehetnek el együtt moziba.

Közös jellemzők

- Több sorozathoz egy sorozatot rendelünk
- Két sorozat elemei közül azokat kell kiválogatnunk, amelyek mindkettőben előfordulnak

Metszet

Megvalósítási ötlet

- Válogassuk ki az egyik sorozat (X) azon elemeit, amelyek a másikban (Y) is előfordulnak
- Ehhez egy kiválogatás és egy eldöntés tételt kell egymásba építeni

Metszet

Bemenet

X:	Egyik feldolgozandó tömb
M:	X elemeinek száma
Y:	Másik feldolgozandó tömb
N:	Y elemeinek száma

Kimenet

Z:	Eredmény tömb
DB:	Metszetbeli elemek száma

Pszeudokód

Eljárás Metszet (X, M, Y, N, Z, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** M-**ig**
    $j \leftarrow 1$
    Ciklus amíg $j \leq N$ **és** $X[i] \neq Y[j]$
      $j \leftarrow j + 1$
    Ciklus vége
    Ha $j \leq N$ **akkor**
      $DB \leftarrow DB + 1$; $Z[DB] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Metszet

Megjegyzések

Kis módosítással a metszet tétel nem csak két sorozat közös elemeinek meghatározására alkalmazható

- Eldöntés: van-e két sorozatnak közös eleme?
- Kiválasztás: adjuk meg a két sorozat egyik közös elemét (ha tudjuk, hogy van ilyen)
- Keresés: ha van, akkor adjuk meg a két sorozat egyik közös elemét
- Megszámolás: hány közös eleme van a két sorozatnak?

Metszet

Keressük meg X egyik olyan elemét, amely benne van Y-ban is

Közös elem keresése

Bemenet: X – egyik tömb
M – X elemeinek száma
Y – másik tömb
N – Y elemeinek száma
Kimenet: VAN – logikai változó; pontosan akkor igaz, ha van a két tömbben közös elem
E – a közös elem értéke (ha van)

Eljárás MetszetbeliElem(X, M, Y, N, VAN, E)
  $i \leftarrow 1$; $VAN \leftarrow$ hamis
  Ciklus amíg $i \leq M$ **és** $\neg VAN$
    $j \leftarrow 1$
    Ciklus amíg $j \leq N$ **és** $X[i] \neq Y[j]$
      $j \leftarrow j + 1$
    Ciklus vége
    Ha $j \leq N$ **akkor**
      $VAN \leftarrow$ igaz; $E \leftarrow X[i]$
    különben
      $i \leftarrow i + 1$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Egyesítés (unió)

Típusfeladatok

1. Két szám prímosztóinak ismeretében adjuk meg legkisebb közös többszörösük prímosztóit.
2. Egy iskola két földrajztanára órarendjének ismeretében adjuk meg azokat az órákat, amikor valamelyikük tud egy órát helyettesíteni.

Közös jellemzőjük

- Két sorozathoz egy sorozatot rendel
- Azokat az elemeket keressük, amelyek a két sorozatból legalább az egyikben benne vannak

Egyesítés (unió)

Bemenet

X:	Egyik feldolgozandó tömb
M:	X elemeinek száma
Y:	Másik feldolgozandó tömb
N:	Y elemeinek száma

Kimenet

Z:	Eredmény tömb
DB:	Unióbeli elemek száma

Pszeudokód

Eljárás Egyesítés(X, M, Y, N, Z, DB)
  $Z \leftarrow X$; $DB \leftarrow M$
  Ciklus $j \leftarrow 1$-**től** N-**ig**
    $i \leftarrow 1$
    Ciklus amíg $i \leq M$ **és** $X[i] \neq Y[j]$
      $i \leftarrow i + 1$
    Ciklus vége
    Ha $i > M$ **akkor**
      $DB \leftarrow DB + 1$; $Z[DB] \leftarrow Y[j]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Egyesítés (unió)

Kis módosítással készíthetünk olyan algoritmust, amely egy sorozatból halmazt készít, azaz egy sorozatot úgy alakít át, hogy az azonos elemek csak egyszer szerepeljenek

Halmazfelsorolás készítés

Bemenet: X – eredeti sorozat
N – X elemeinek száma
Kimenet: Z – halmazzá alakított sorozat
DB – Z elemeinek száma

Eljárás HalmazfelsorolásKészítés(X, N, Z, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow 1$-**től** N-**ig**
    $j \leftarrow 1$
    Ciklus amíg $j \leq DB$ **és** $X[i] \neq Z[j]$
      $j \leftarrow j + 1$
    Ciklus vége
    Ha $j > DB$ **akkor**
      $DB \leftarrow DB + 1$
      $Z[DB] \leftarrow X[i]$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Összefuttatás (rendezettek uniója)

Típusfeladatok

1. Egy osztály lány-, illetve fiú tanulóinak névsora alapján állítsuk elő az osztálynévsort.
2. Egy iskolában négy szakkörre járnak tanulók (van aki többre is). A szakkörnévsorok alapján állítsuk elő a szakkörre járó tanulók névsorát.

Közös jellemzők

- Az általános egyesítéshez képest itt specialitás, hogy mindegyik sorozat rendezett

Összefuttatás

Bemenet

X:	Egyik feldolgozandó tömb
M:	X elemeinek száma
Y:	Másik feldolgozandó tömb
N:	Y elemeinek száma

Kimenet

Z:	Eredmény tömb
DB:	Unióbeli elemek száma

Pszeudokód

Eljárás Összefuttatás(X, M, Y, N, Z, DB)
  $i \leftarrow 1$; $j \leftarrow 1$; $DB \leftarrow 0$
  Ciklus amíg $i \leq M$ **és** $j \leq N$
    $DB \leftarrow DB + 1$
    Elágazás
      $X[i] < Y[j]$ esetén $Z[DB] \leftarrow X[i]$; $i \leftarrow i + 1$
      $X[i] = Y[j]$ esetén $Z[DB] \leftarrow X[i]$; $i \leftarrow i + 1$; $j \leftarrow j + 1$
      $X[i] > Y[j]$ esetén $Z[DB] \leftarrow Y[j]$; $j \leftarrow j + 1$
    Elágazás vége
  Ciklus vége
  Ciklus amíg $i \leq M$
    $DB \leftarrow DB + 1$; $Z[DB] \leftarrow X[i]$; $i \leftarrow i + 1$
  Ciklus vége
  Ciklus amíg $j \leq N$
    $DB \leftarrow DB + 1$; $Z[DB] \leftarrow Y[j]$; $j \leftarrow j + 1$
  Ciklus vége
Eljárás vége

Összefuttatás (rendezettek uniója)

Ha $X[M] = Y[N]$, akkor az utolsó két ciklusra nincs szükség. Ezt a helyzetet magunk is előidézhetjük.

Pszeudokód

Eljárás Összefuttatás(X, M, Y, N, Z, DB)
  $i \leftarrow 1$; $j \leftarrow 1$; $DB \leftarrow 0$
  $X[M+1] \leftarrow +\infty$; $Y[N+1] \leftarrow +\infty$
  Ciklus amíg $i < M+1$ **vagy** $j < N+1$
    $DB \leftarrow DB + 1$
    Elágazás
      $X[i] < Y[j]$ esetén $Z[DB] \leftarrow X[i]$; $i \leftarrow i+1$
      $X[i] = Y[j]$ esetén $Z[DB] \leftarrow X[i]$; $i \leftarrow i+1$; $j \leftarrow j+1$
      $X[i] > Y[j]$ esetén $Z[DB] \leftarrow Y[j]$; $j \leftarrow j+1$
    Elágazás vége
  Ciklus vége
Eljárás vége

Összefuttatás (rendezettek uniója)

Ha nincs a két sorozatban azonos elem, akkor a megvalósítás még egyszerűbb.

Pszeudokód

Eljárás Összefuttatás(X, M, Y, N, Z, DB)
  $i \leftarrow 1$; $j \leftarrow 1$; $DB \leftarrow 0$
  $X[M+1] \leftarrow +\infty$; $Y[N+1] \leftarrow +\infty$
  Ciklus amíg $i < M+1$ **vagy** $j < N+1$
    $DB \leftarrow DB + 1$
    Ha $X[i] < Y[j]$ **akkor**
      $Z[DB] \leftarrow X[i]$; $i \leftarrow i+1$
    különben
      $Z[DB] \leftarrow Y[j]$; $j \leftarrow j+1$
    Elágazás vége
  Ciklus vége
Eljárás vége

Tartalom

Érték- és referencia típusú változók

Érték típusú változók

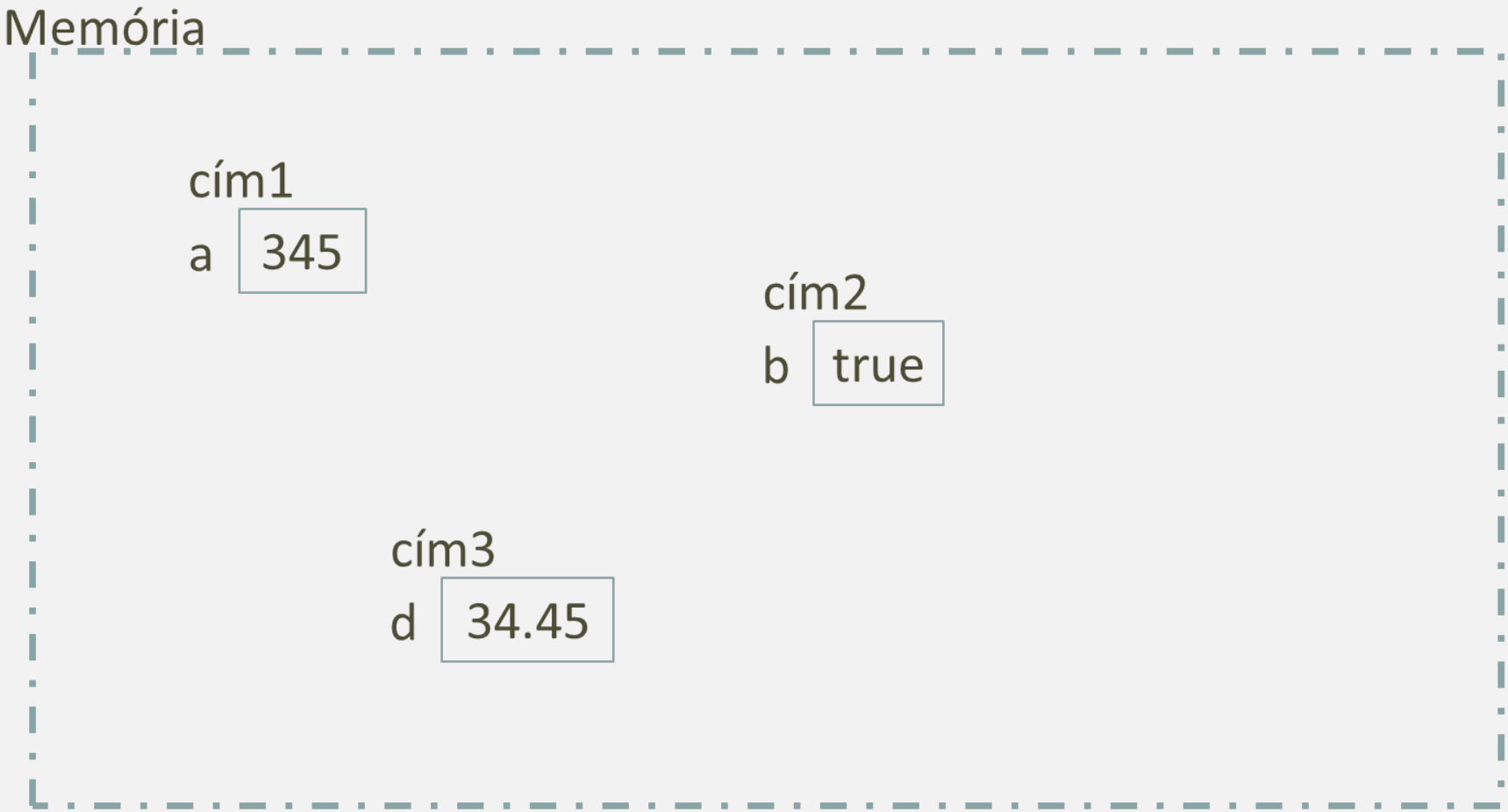
- Konkrét értéket tárolnak
- Az `a = b` értékadáskor a `b` értéke átmásolódik `a`-ba
- Egész, valós, logikai, karakter, felsorolás, struktúra típusok

Referencia típusú változók

- Csak egy memória helyre tárolnak hivatkozást, ahol a konkrét érték található
- Az `a = b` értékadáskor a memóriabeli cím másolódik $\rightarrow$ `b` ugyanazon memóriacímen lévő értékre fog hivatkozni, mint amire `a` hivatkozik
- Tömb, osztály, interfész, delegált, esemény típusok

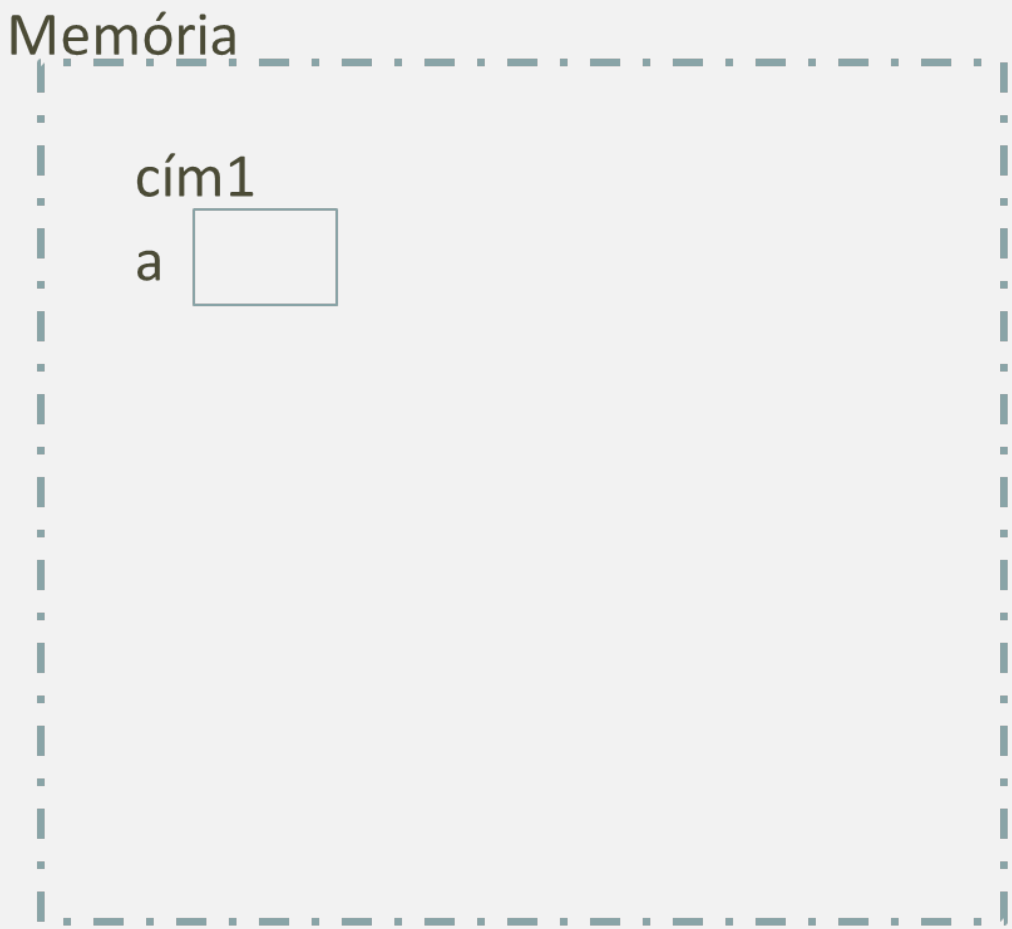
Változók a memóriában

Sematikus ábra



Érték típusú változók a memóriában (1)

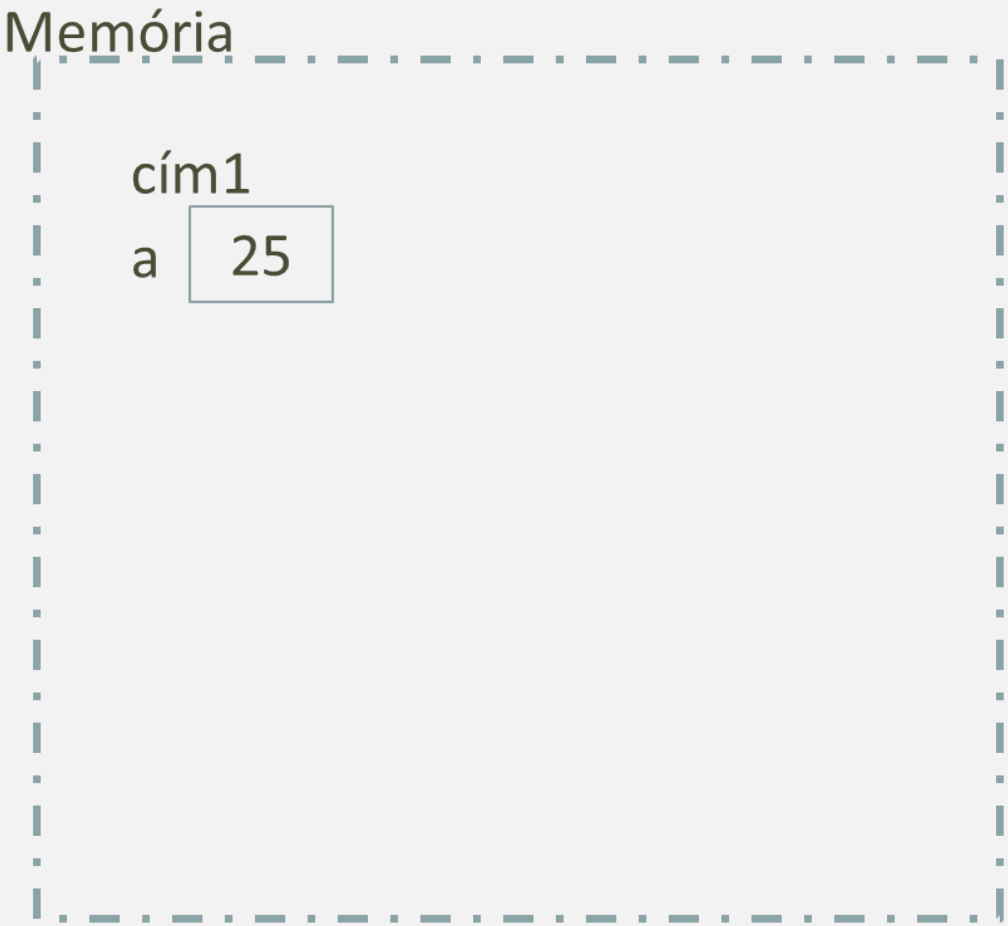
Sematikus ábra



int a;

Érték típusú változók a memóriában (2)

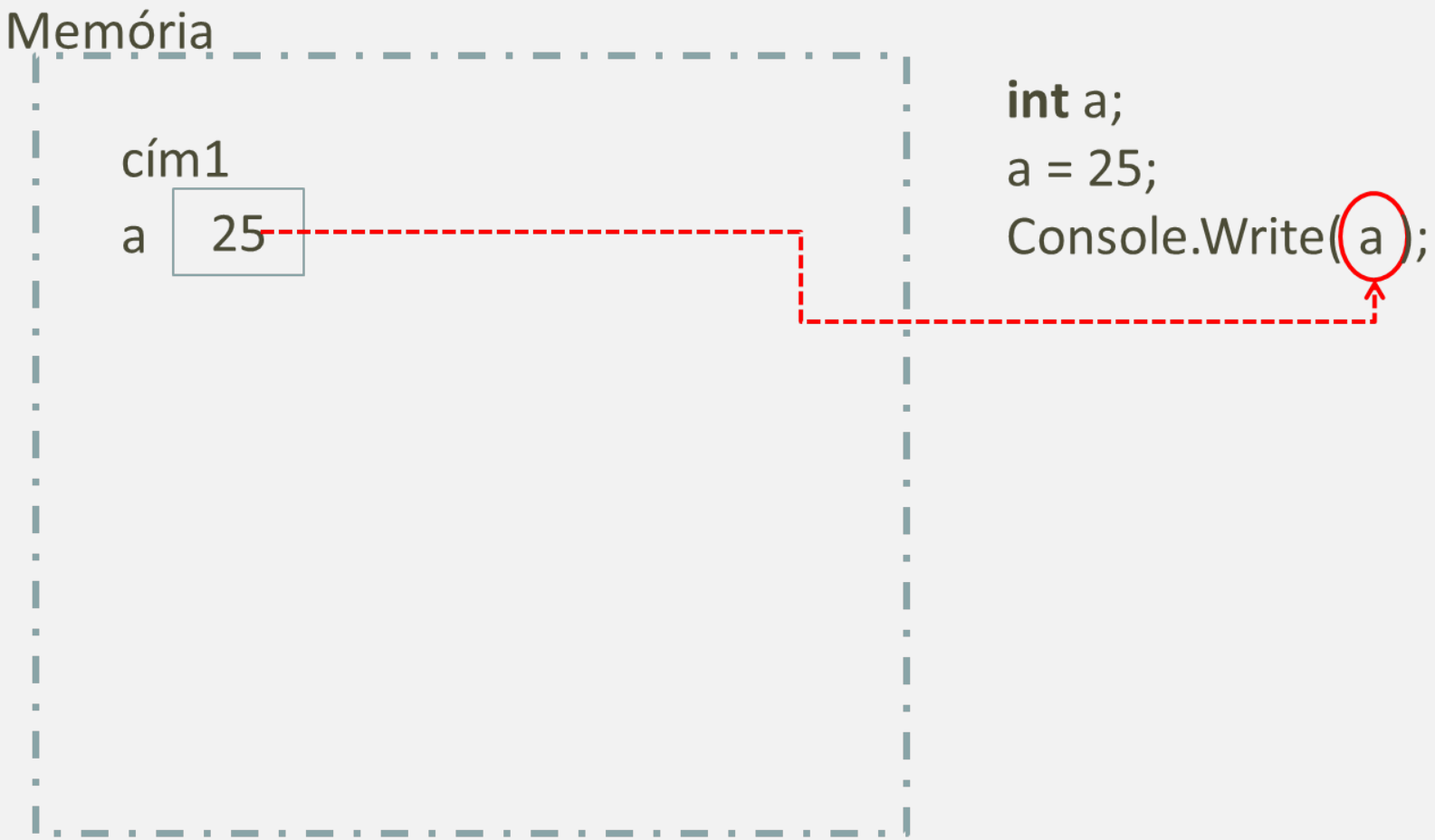
Sematikus ábra




```
int a;
a = 25;
```

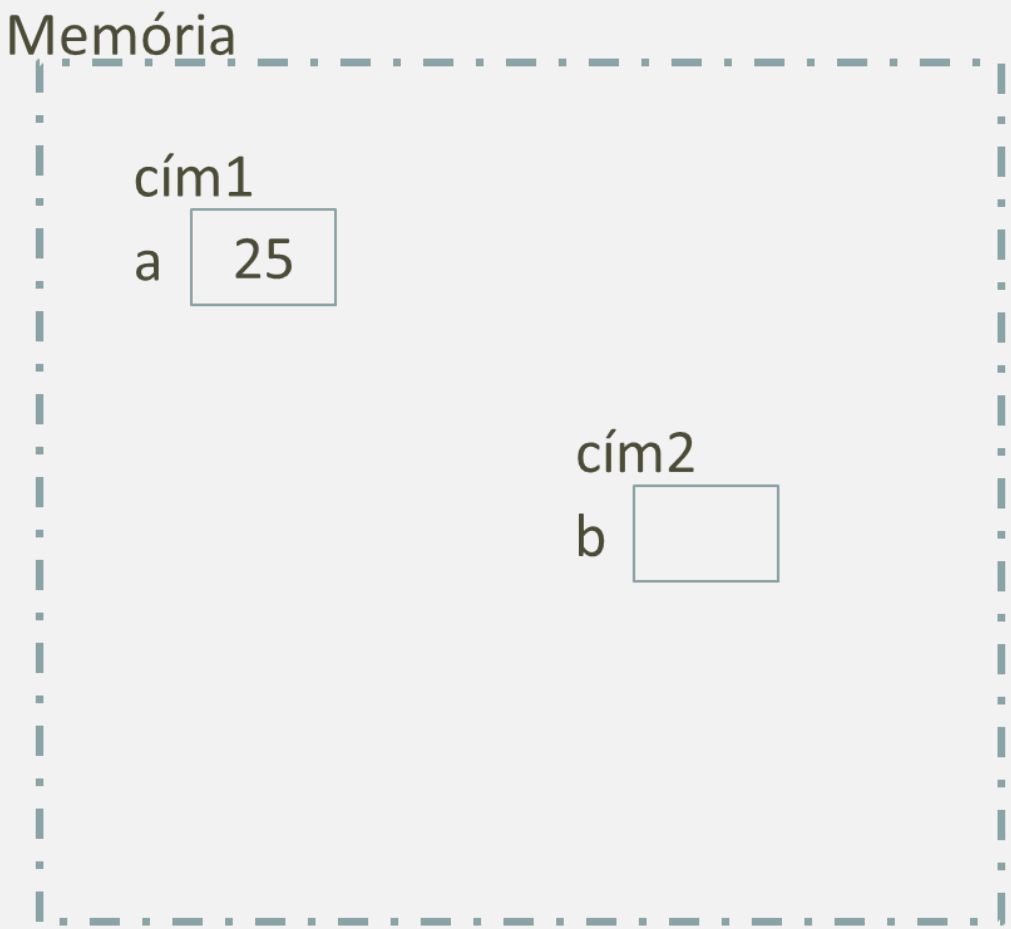
Érték típusú változók a memóriában (3)

Sematikus ábra



Érték típusú változók a memóriában (4)

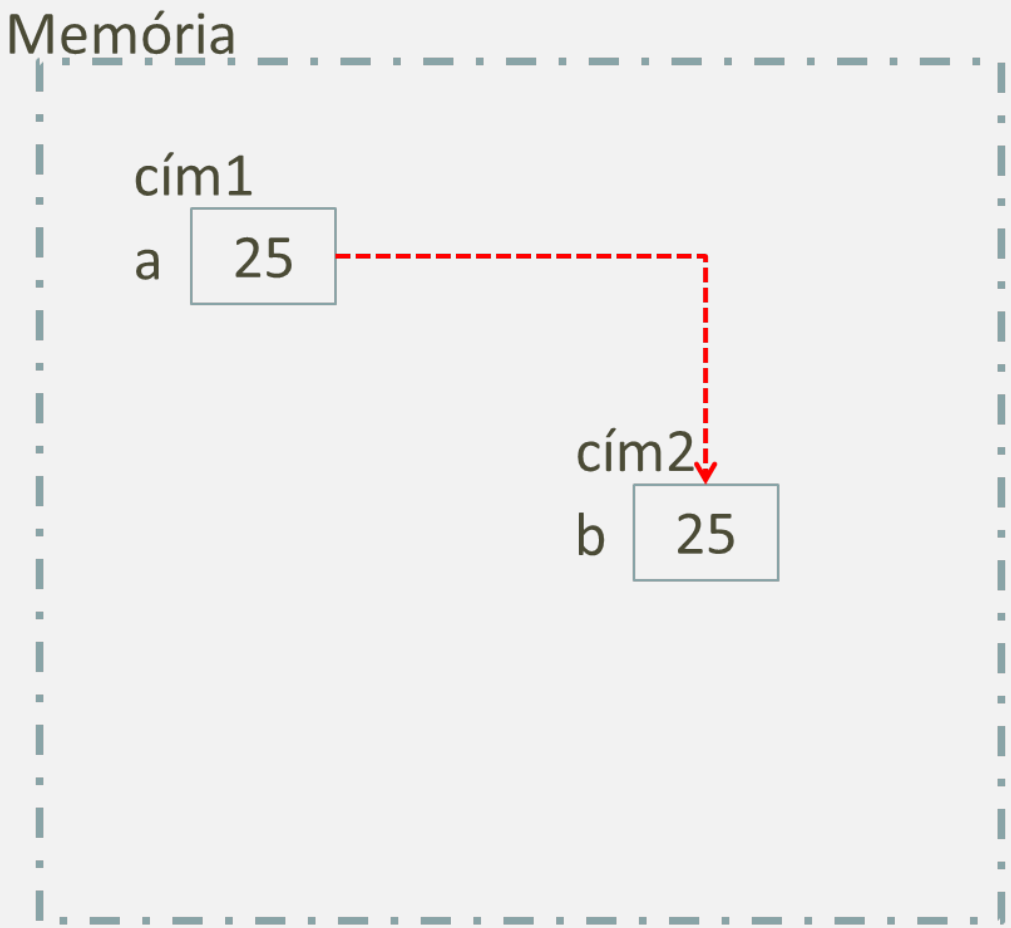
Sematikus ábra




```
int a;
a = 25;
Console.Write( a );
int b;
```

Érték típusú változók a memóriában (5)

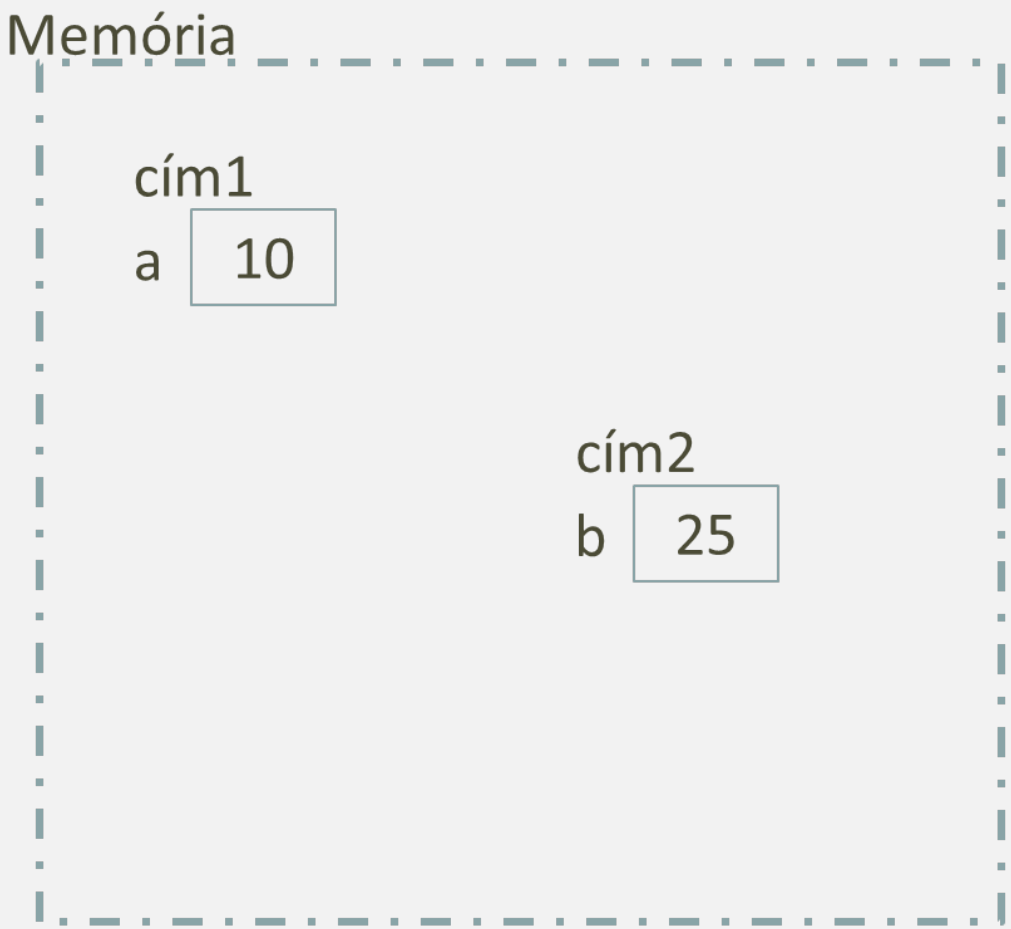
Sematikus ábra




```
int a;
a = 25;
Console.Write( a );
int b;
b = a;
```

Érték típusú változók a memóriában (6)

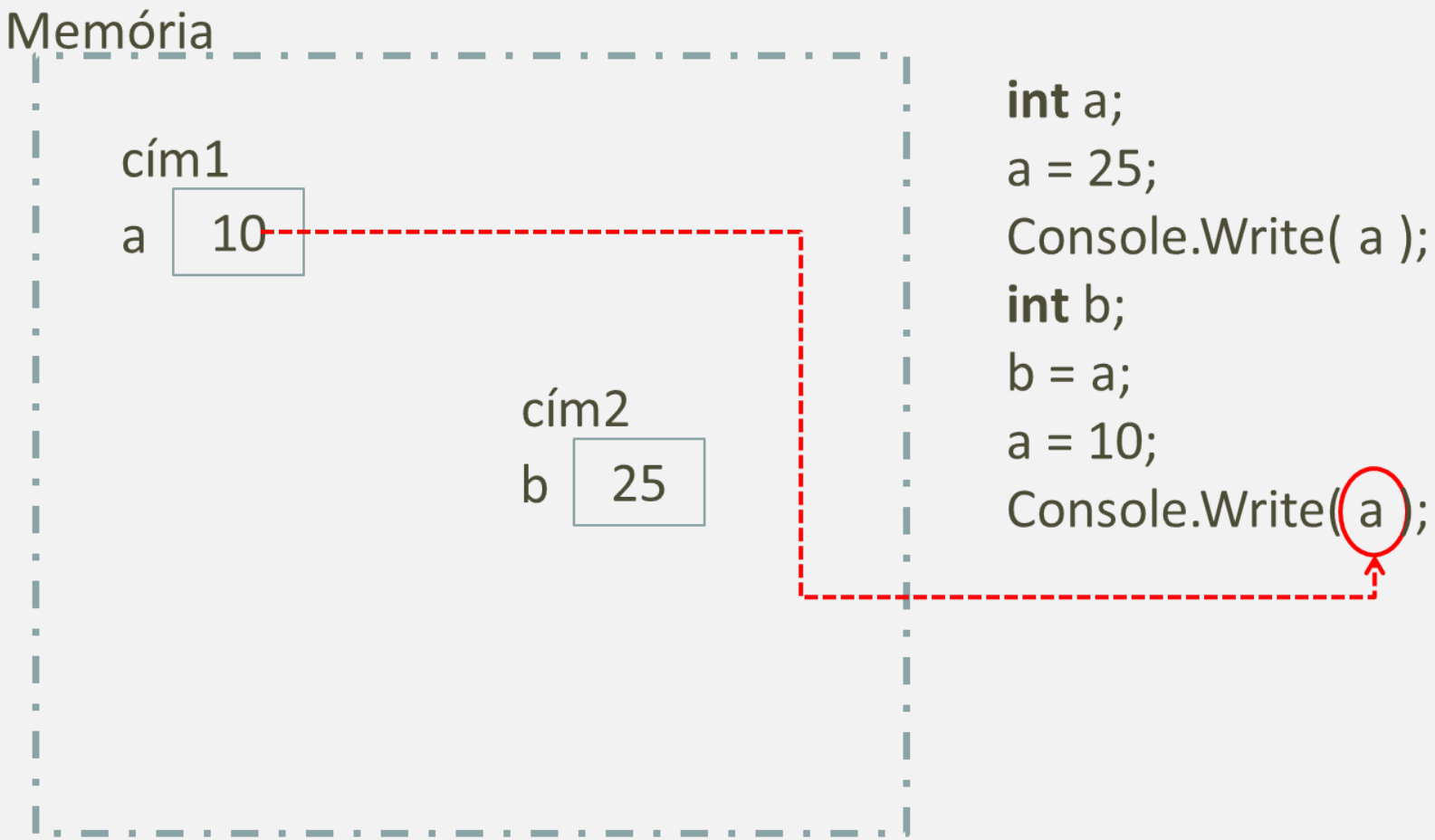
Sematikus ábra




```
int a;
a = 25;
Console.Write( a );
int b;
b = a;
a = 10;
```

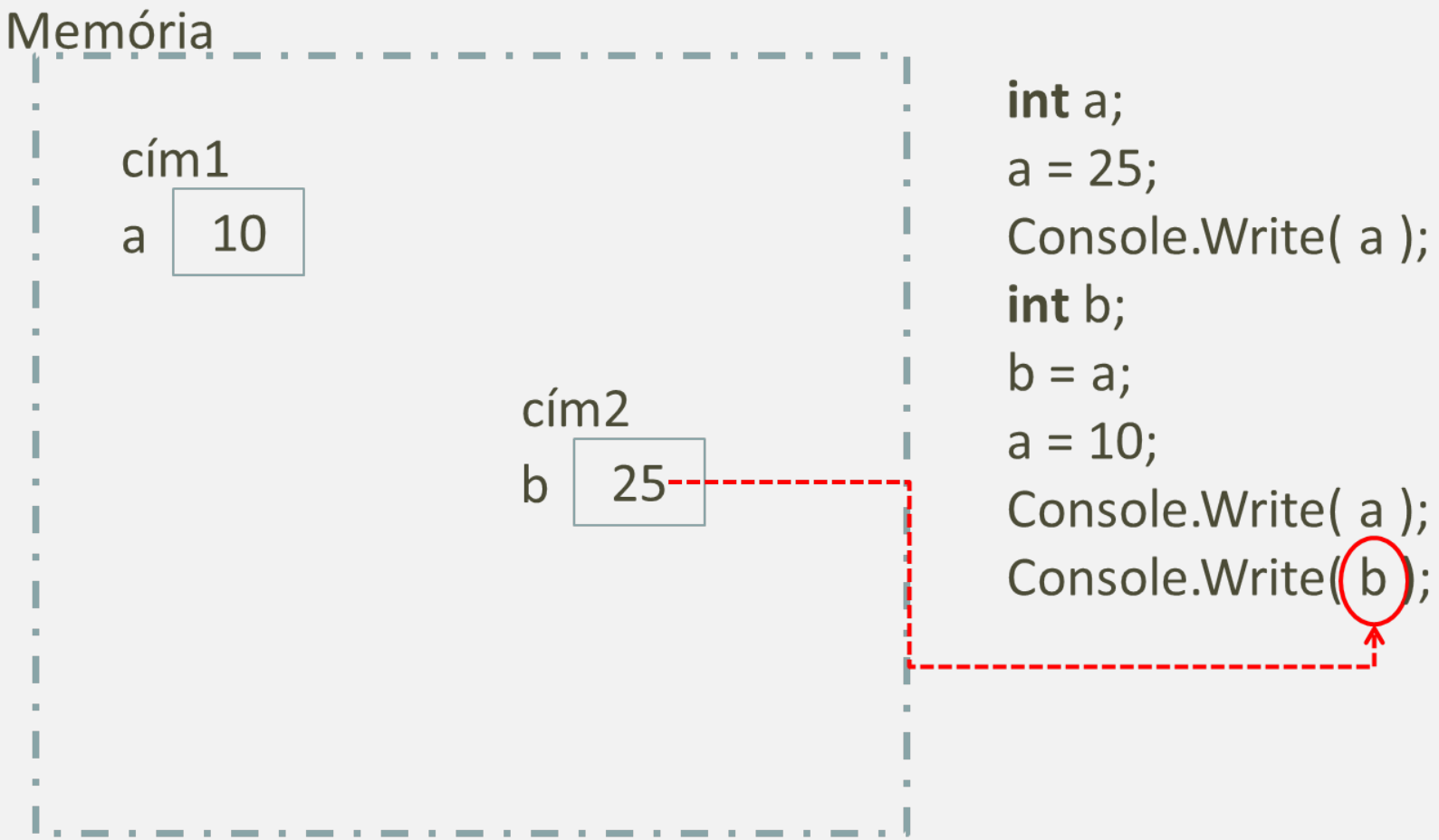
Érték típusú változók a memóriában (7)

Sematikus ábra



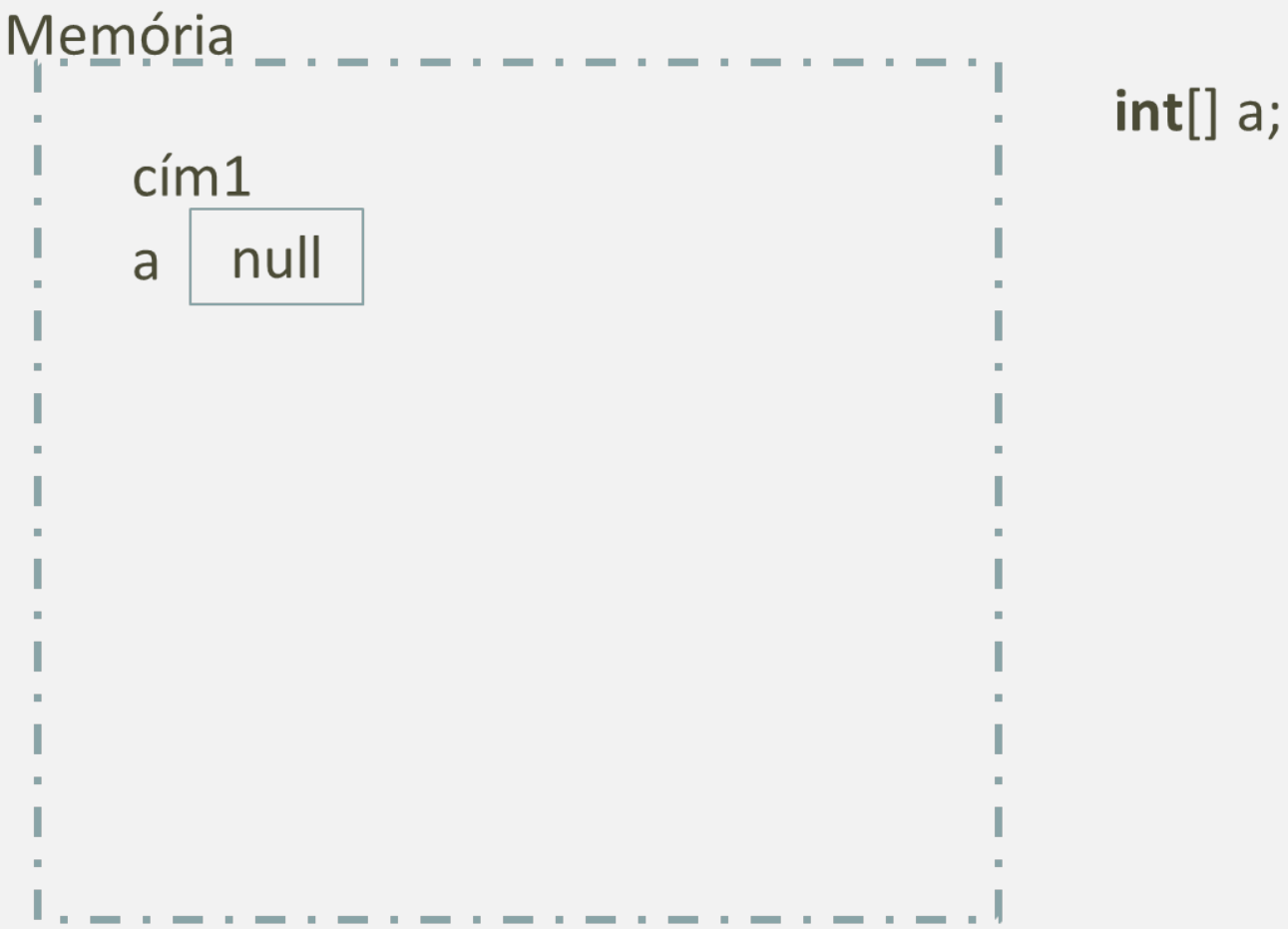
Érték típusú változók a memóriában (8)

Sematikus ábra



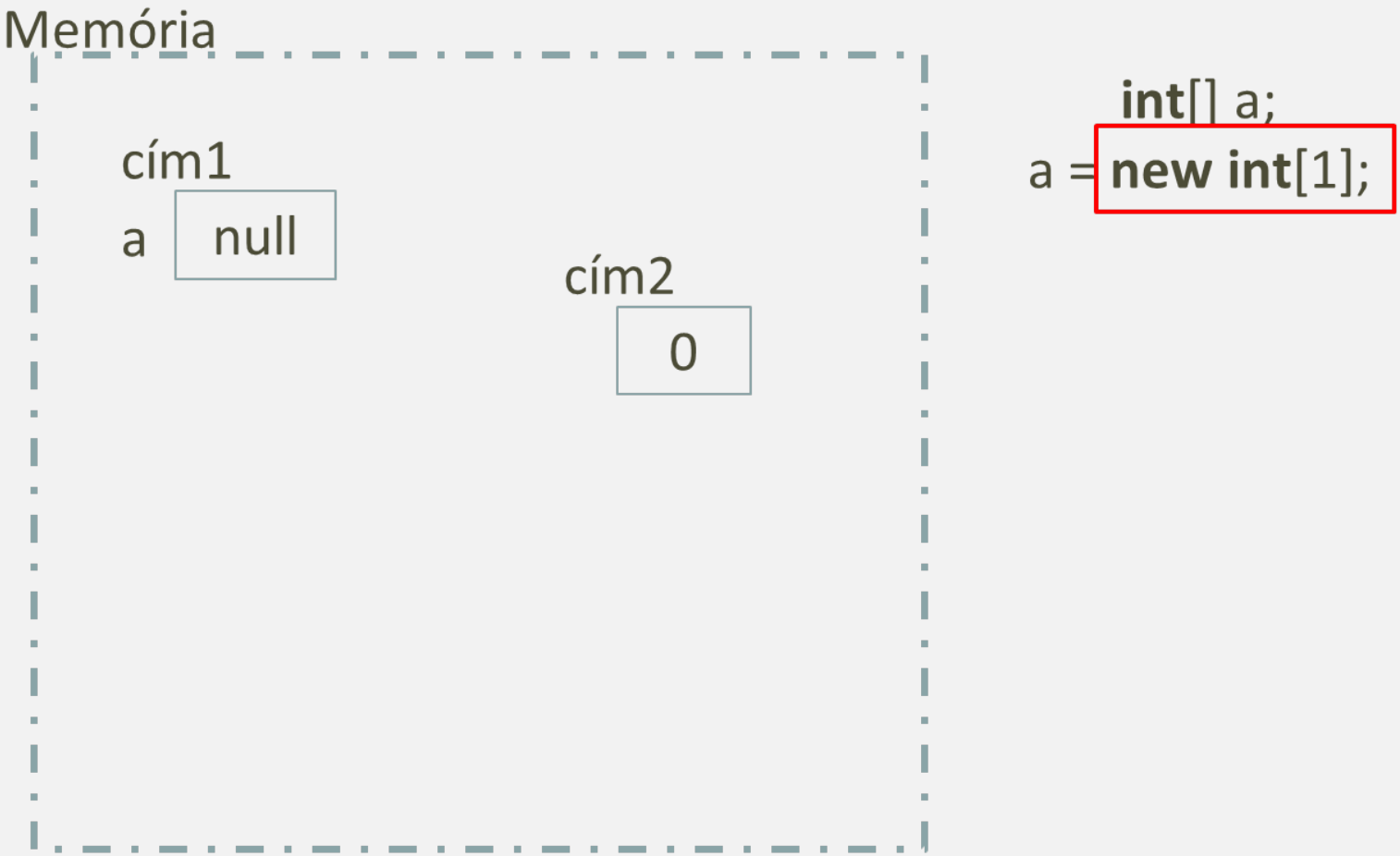
Referencia típusú változók a memóriában (1)

Sematikus ábra



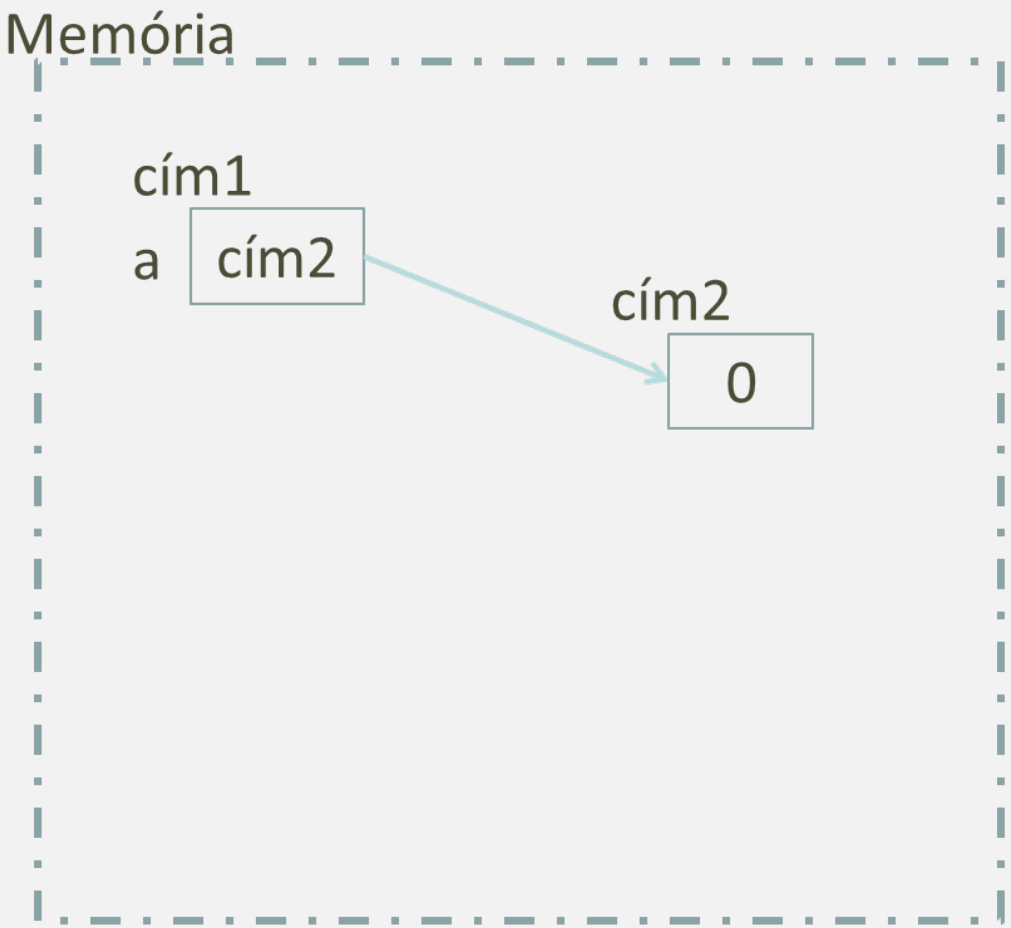
Referencia típusú változók a memóriában (2)

Sematikus ábra



Referencia típusú változók a memóriában (3)

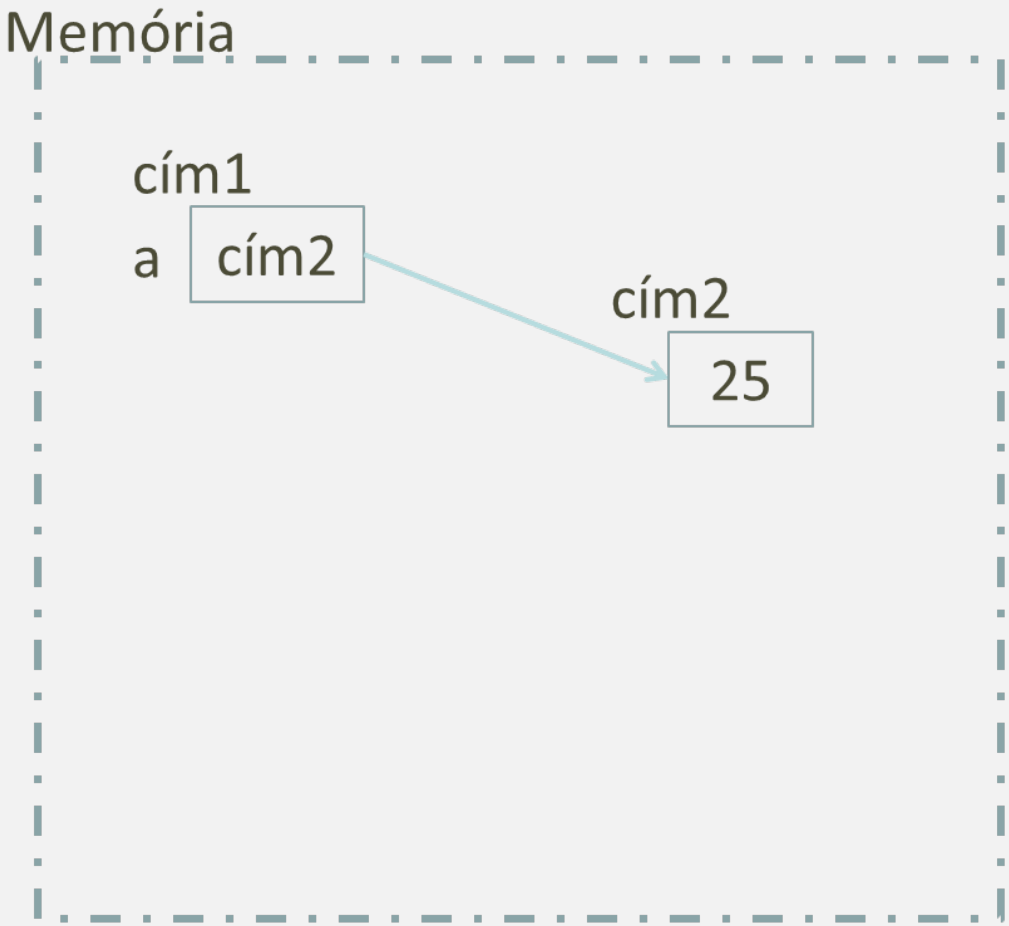
Sematikus ábra




```
int[] a;
a = new int[1];
```

Referencia típusú változók a memóriában (4)

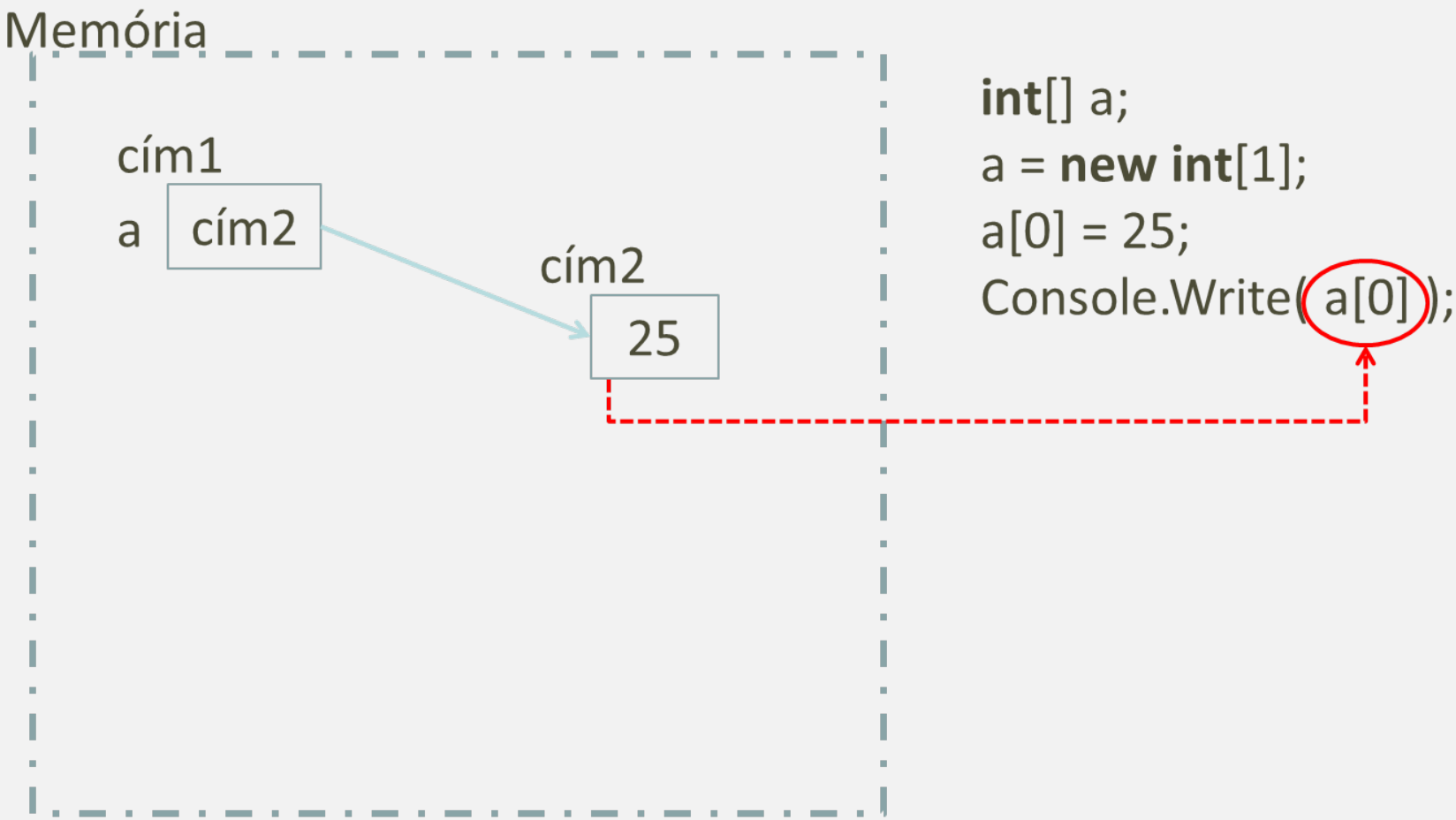
Sematikus ábra




```
int[] a;
a = new int[1];
a[0] = 25;
```

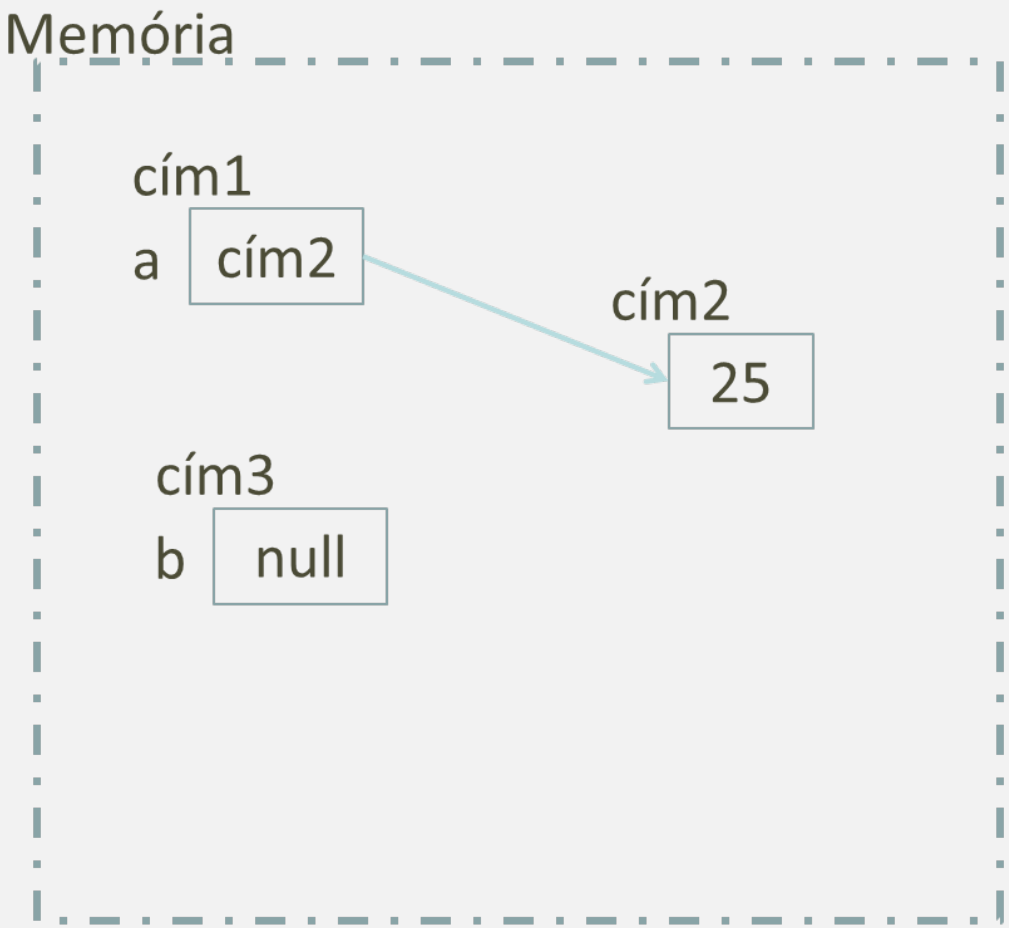
Referencia típusú változók a memóriában (5)

Sematikus ábra



Referencia típusú változók a memóriában (6)

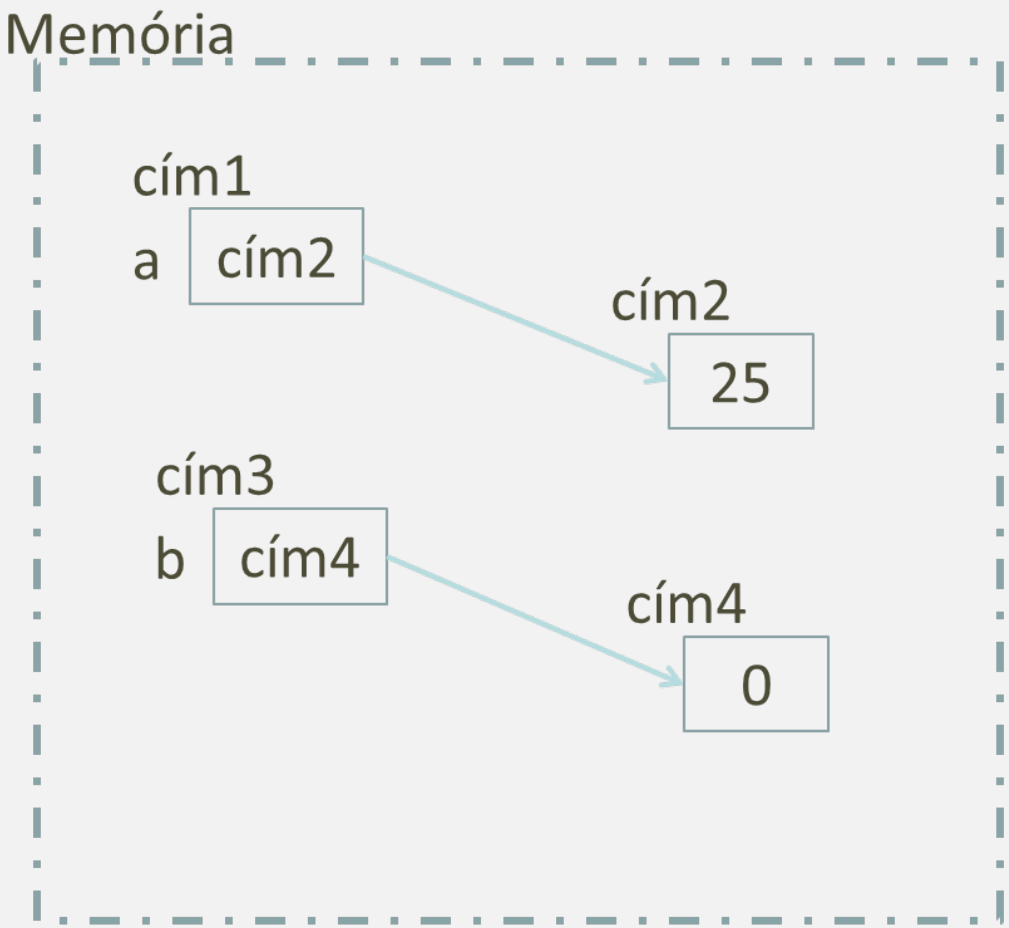
Sematikus ábra




```
int[] a;
a = new int[1];
a[0] = 25;
Console.Write( a[0] );
int[] b;
```

Referencia típusú változók a memóriában (7)

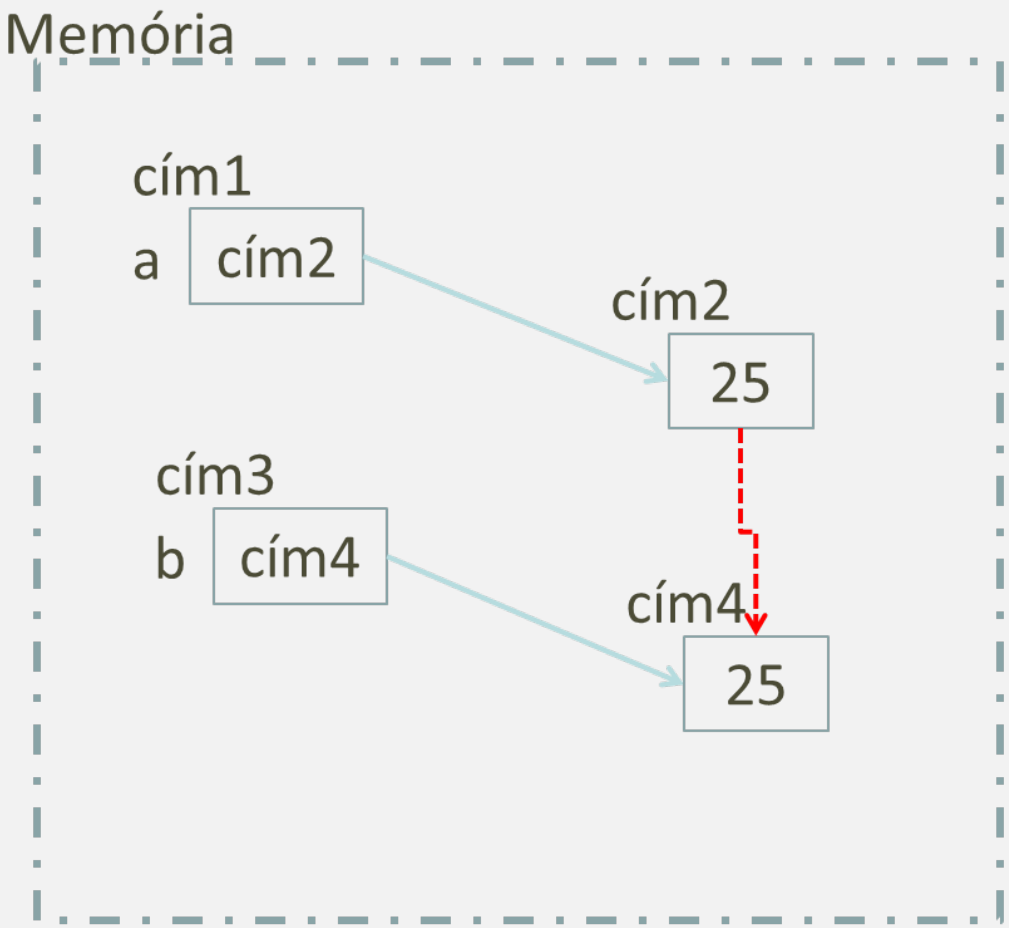
Sematikus ábra




```
int[] a;
a = new int[1];
a[0] = 25;
Console.Write( a[0] );
int[] b;
b = new int[1];
```

Referencia típusú változók a memóriában (8)

Sematikus ábra

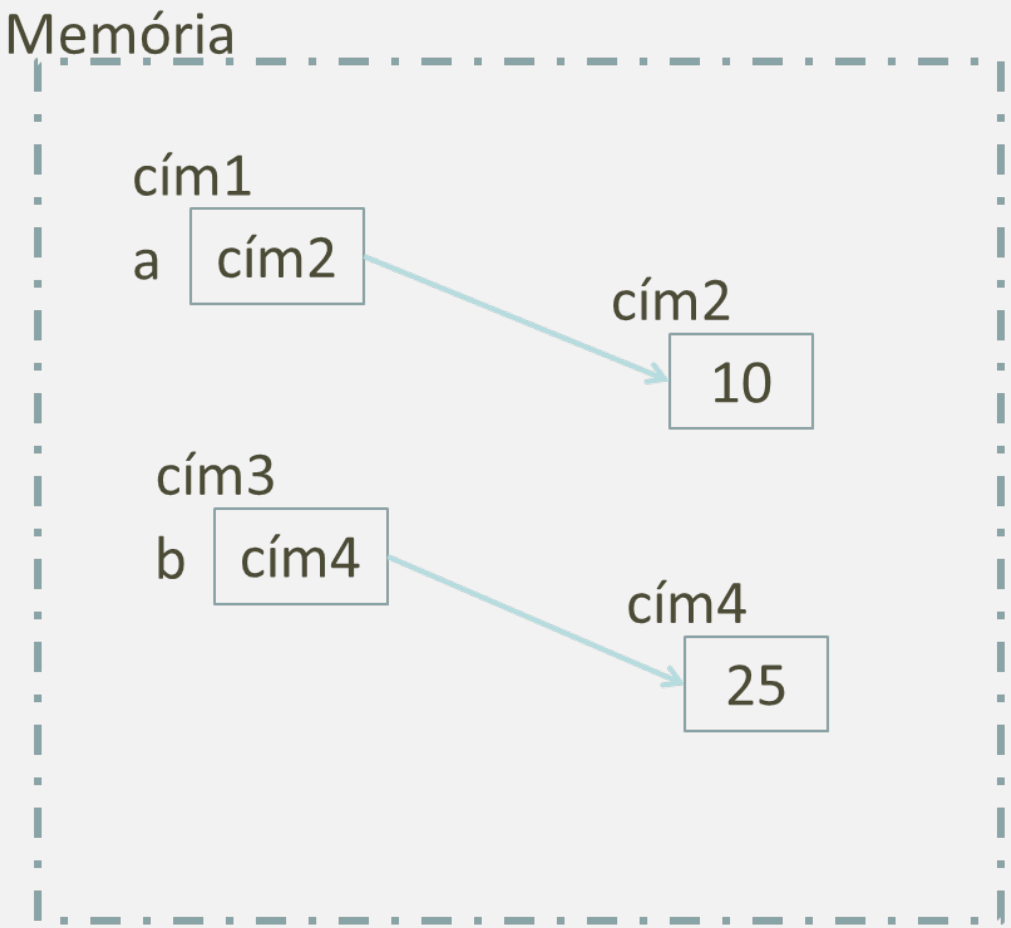



```
int[] a;
a = new int[1];
a[0] = 25;
Console.Write( a[0] );
int[] b;
b = new int[1];
b[0] = a[0];
```

Ez még érték típusú változók értékadása!

Referencia típusú változók a memóriában (9)

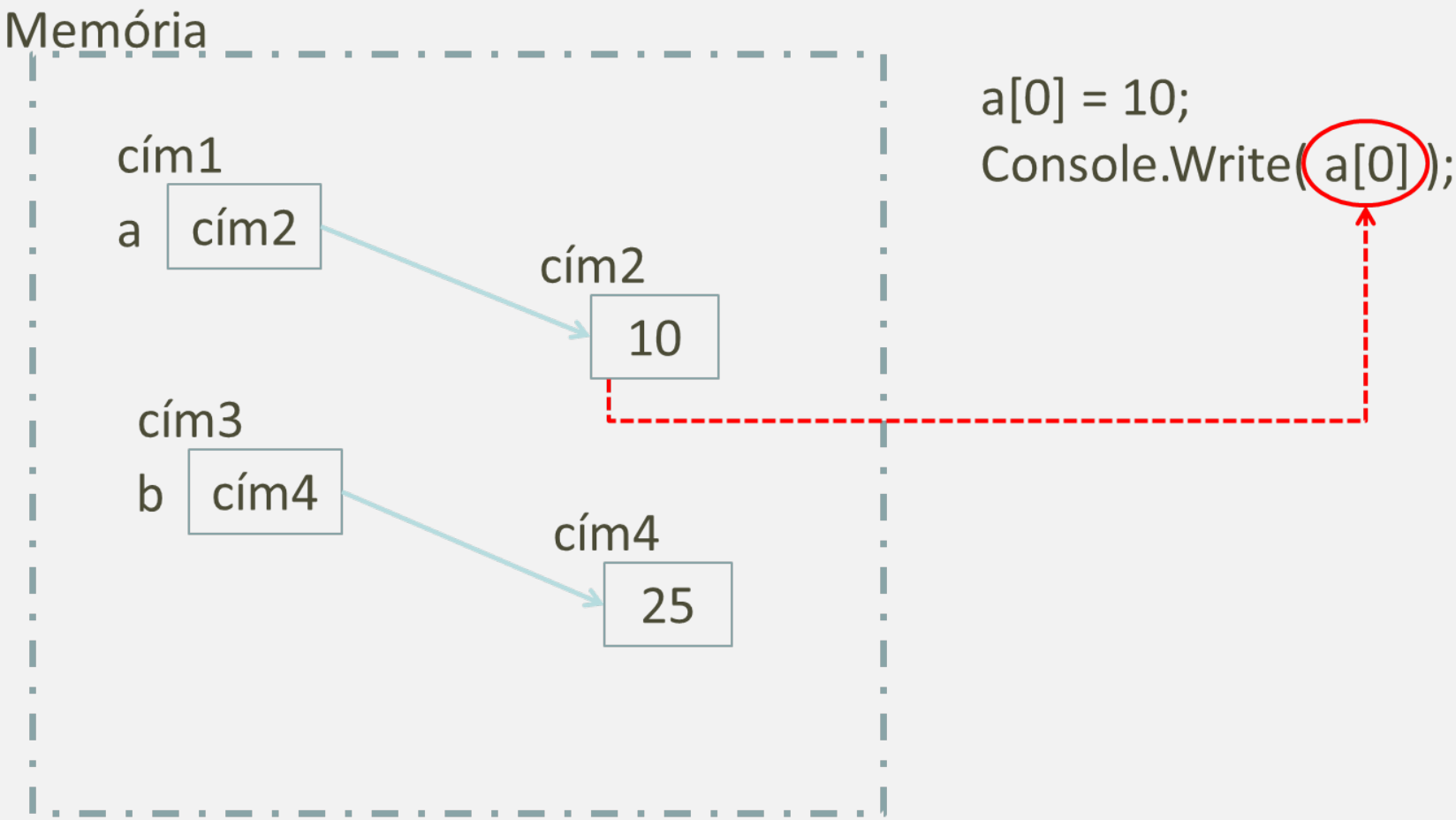
Sematikus ábra



a[0] = 10;

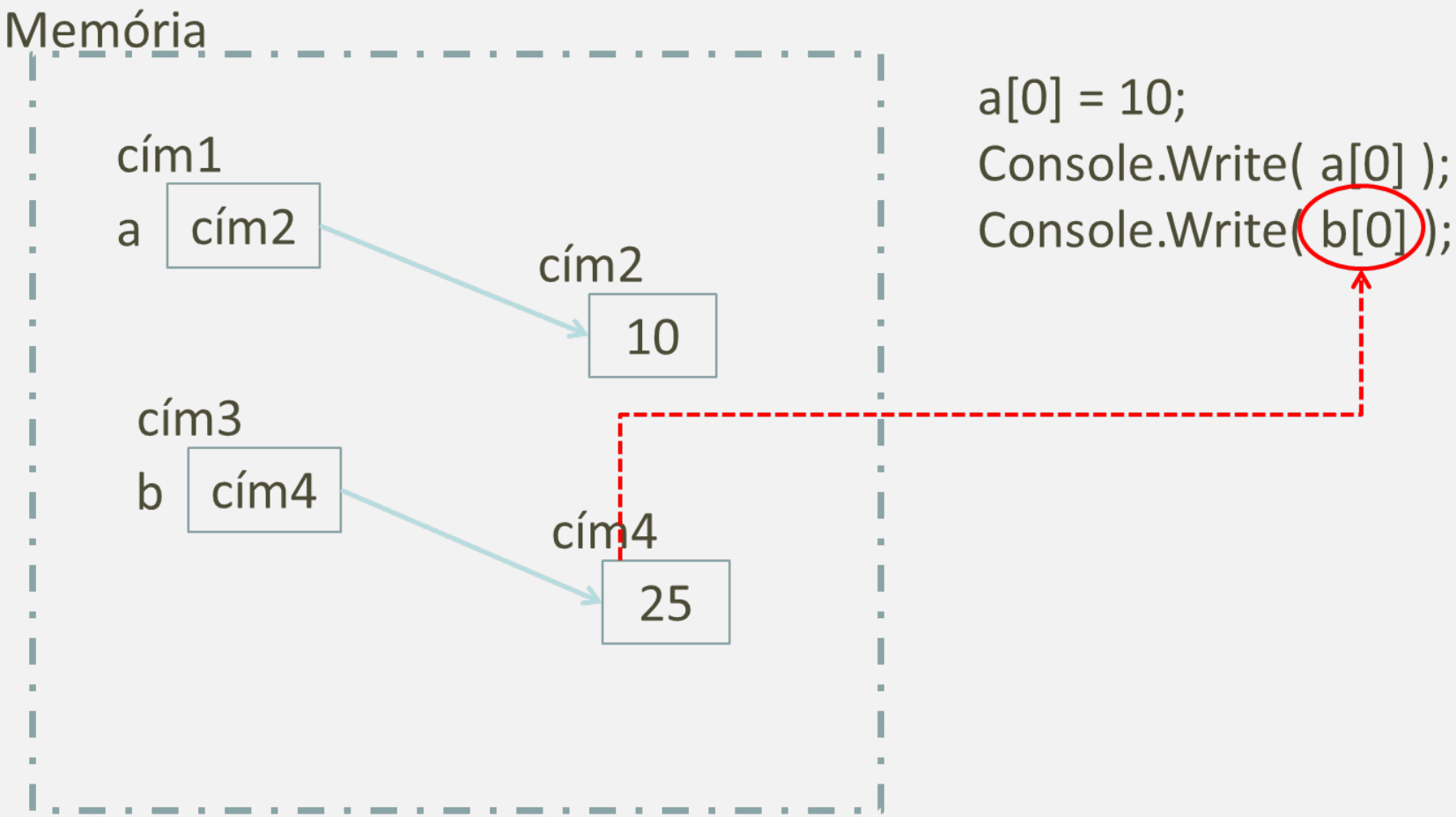
Referencia típusú változók a memóriában (10)

Sematikus ábra



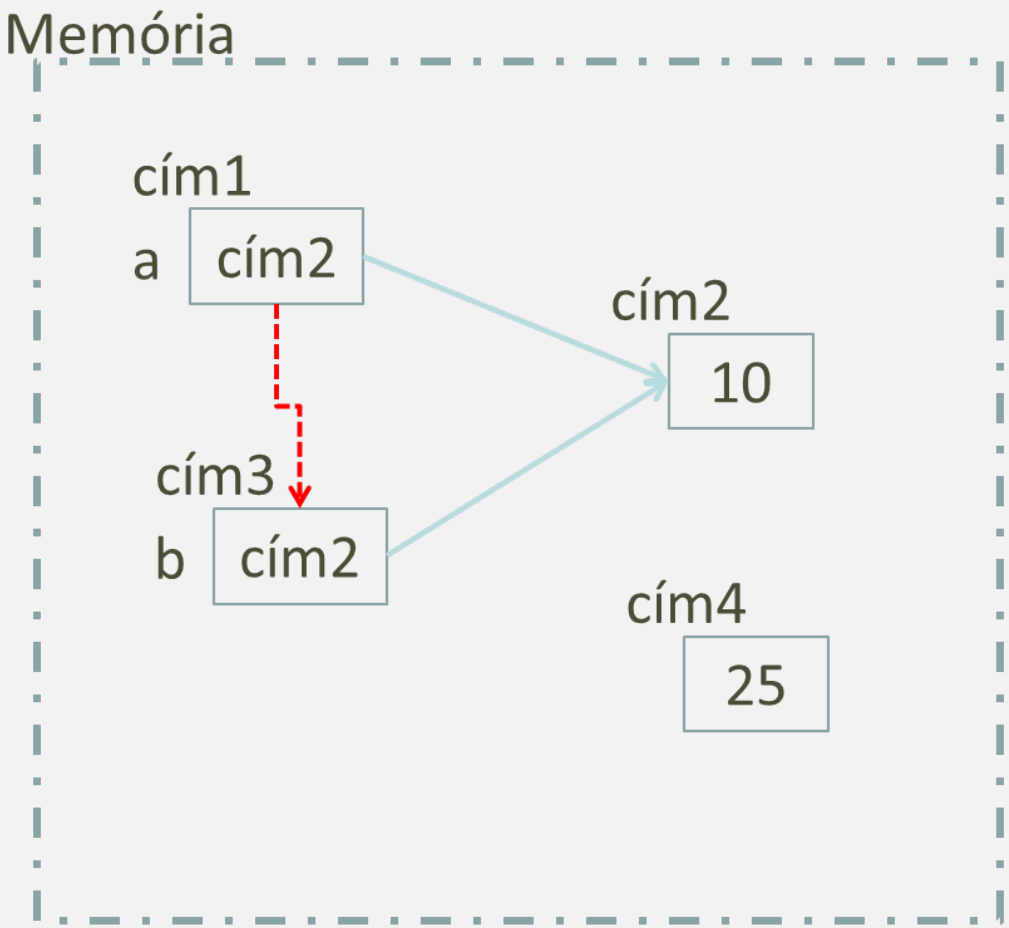
Referencia típusú változók a memóriában (11)

Sematikus ábra



Referencia típusú változók a memóriában (12)

Sematikus ábra

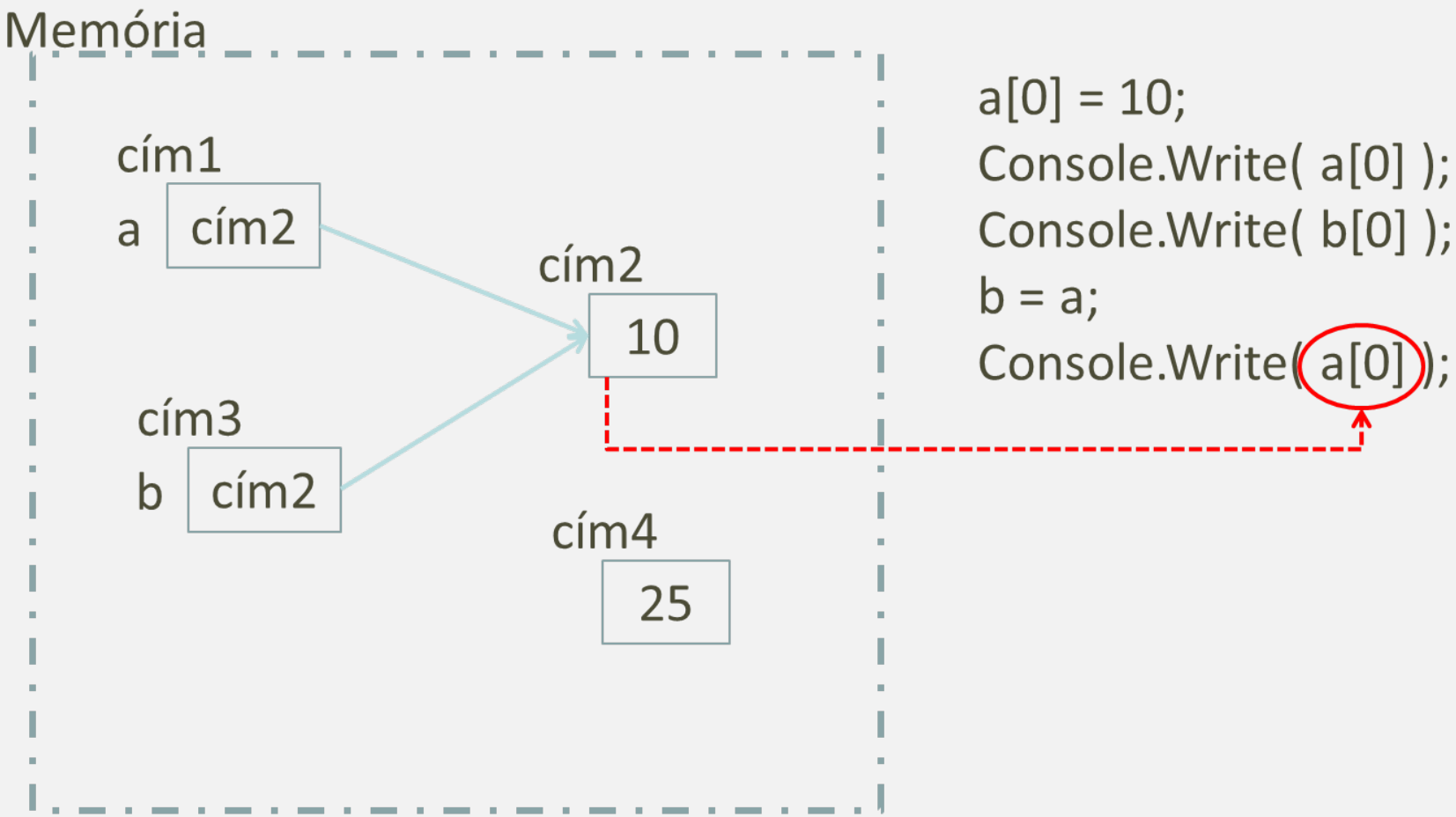



```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
```

Referencia típusú változók értékadása!

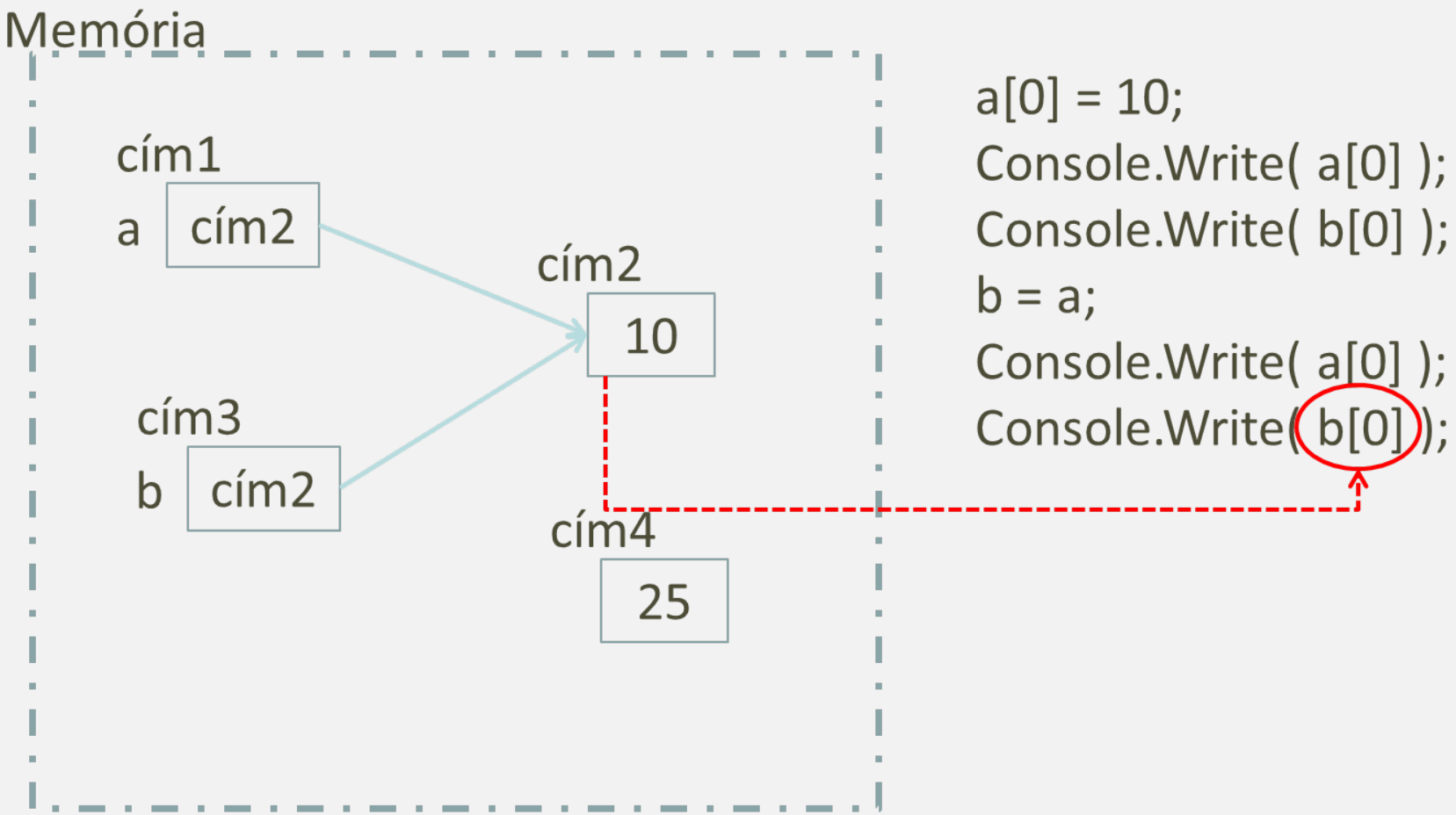
Referencia típusú változók a memóriában (13)

Sematikus ábra



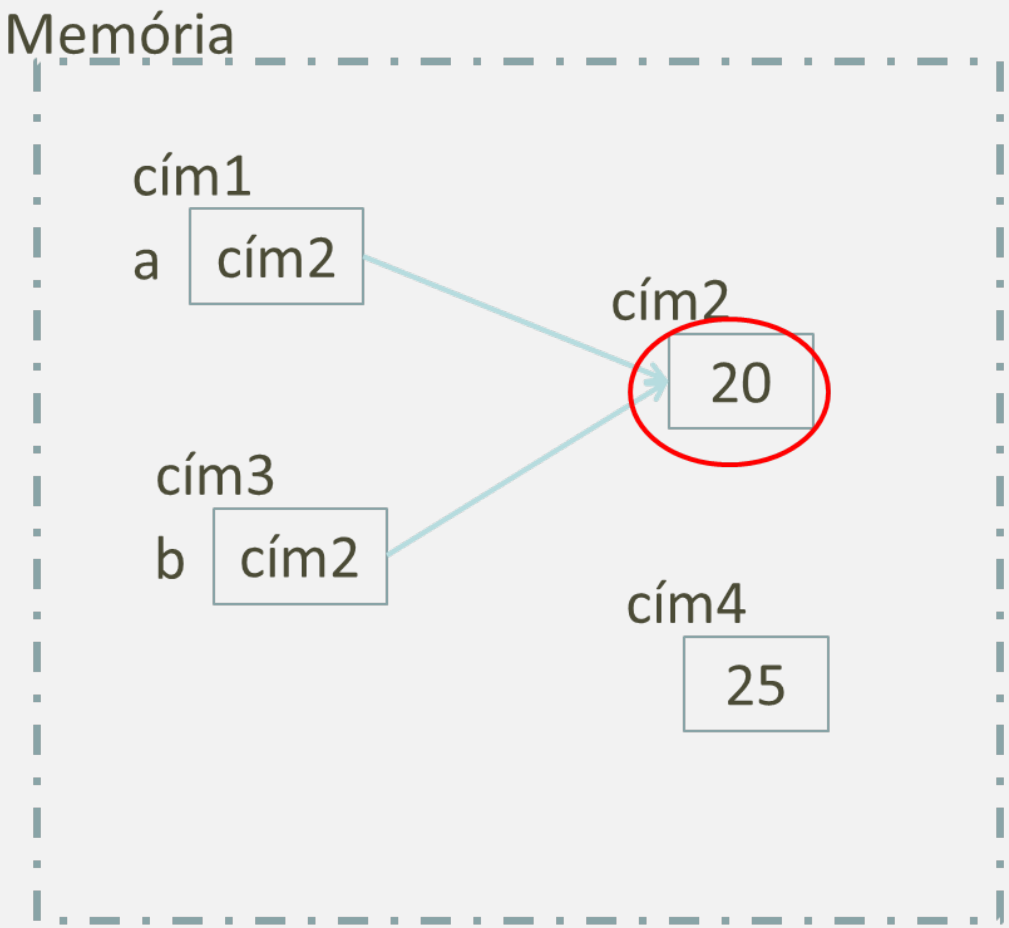
Referencia típusú változók a memóriában (14)

Sematikus ábra



Referencia típusú változók a memóriában (15)

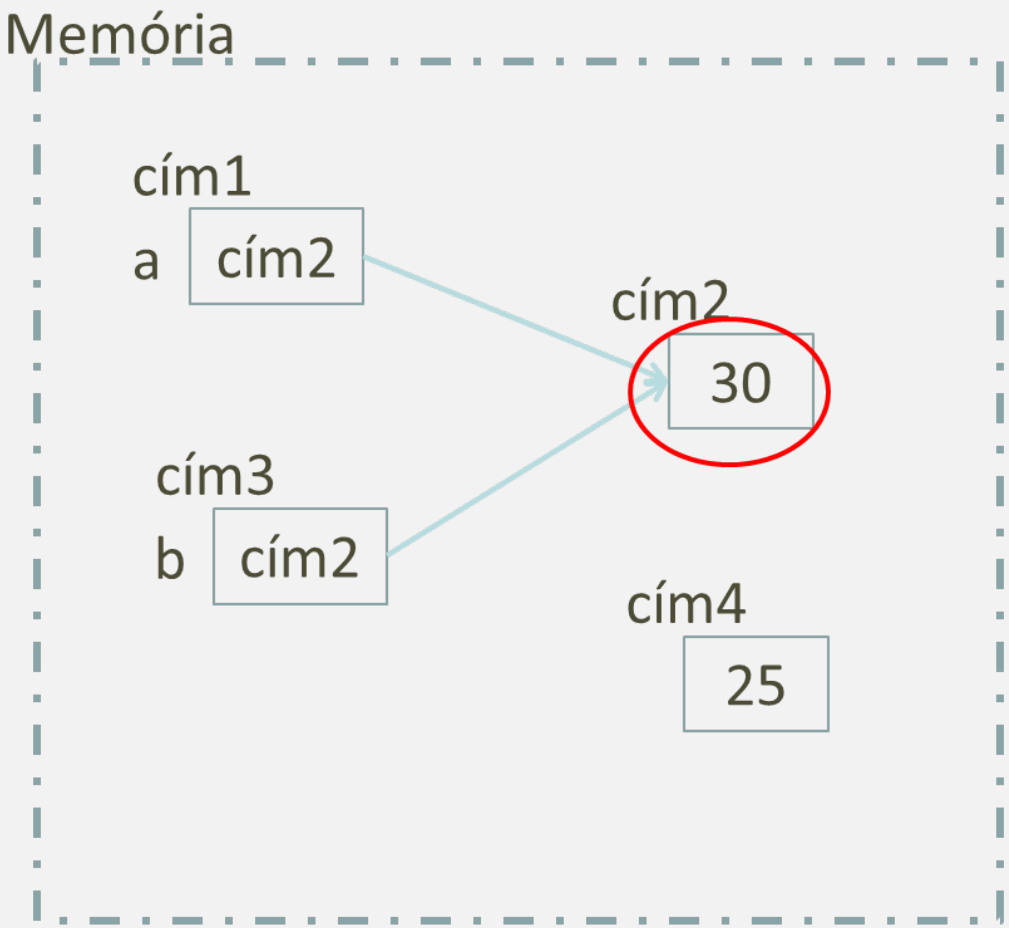
Sematikus ábra




```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
Console.Write( a[0] );
Console.Write( b[0] );
a[0] = 20;
```

Referencia típusú változók a memóriában (16)

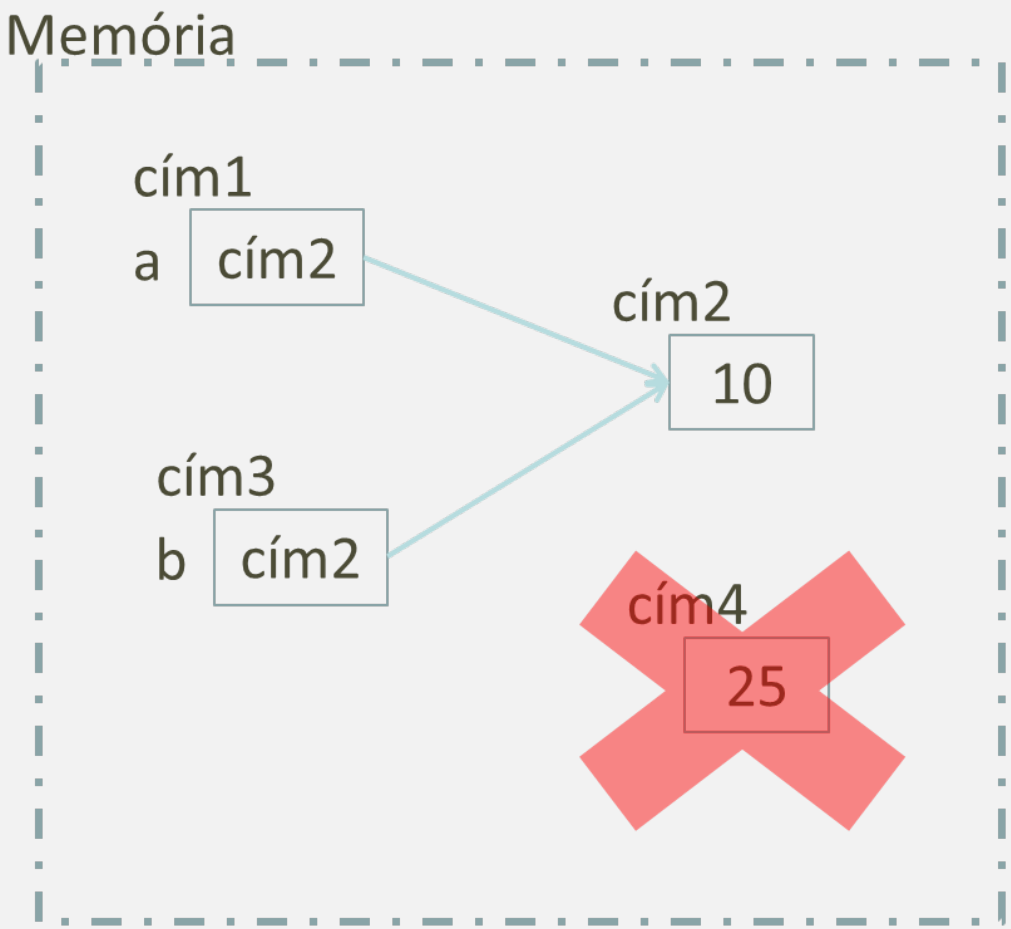
Sematikus ábra




```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
Console.Write( a[0] );
Console.Write( b[0] );
a[0] = 20;
b[0] = 30;
```

Referencia típusú változók a memóriában (17)

Sematikus ábra



Azt a memóriaterületet, amelyre egyetlen referenciaváltozó sem hivatkozik, a GC automatikusan felszabadítja

Tartalom

Metódusok

- A metódus egy kódblokk, amely utasítások sorozatát tartalmazza
- A program azáltal futtatja ezt a kódblokkot, hogy meghívja a metódust és megszabja a szükséges paramétereit
- C#-ban minden futtatandó utasítás egy metódusban helyezkedik el
  - Eddig programjainkat a `Main()` metódusba írtuk
- A többször használt kódrészeket írjuk metódusba
  - "Copy-Paste" helyett
  - Célszerű a hosszú metódusok feldarabolása az egyszerűbb értelmezés céljából

Metódusok

Szintaktika

```
visszatérési_típus metódusnév(paraméterek)
{metódustörzs}
```

Metódusok típusa

- A típus a visszaadott érték típusa vagy `void`, ha nem adunk vissza semmit
- Egy metódusnak legfeljebb egy visszatérési értéke van
  - de az lehet tömb is
- A visszatérési érték típus előtt állhatnak különféle módosítók
  - Pl. `static`, `abstract`, `override`, `new`, illetve láthatóságot jelző kulcsszavak

Metódustörzs

- A metódushoz tartozó utasítások, amelyek használhatják a metódusnak átadott paramétereket
- A metódus visszatérési értékét a `return` kulcsszó után adjuk meg. Ennek hatására a metódusból azonnal visszatérünk a hívóhoz, akkor is, ha még vannak további utasítások a `return` után.
- Ha a metódus több ágon is véget érhet, akkor minden ág végére kell `return`
- Visszatérési érték nélküli (`void`) metódusnál – ha a program mindig a metódustörzs fizikai végénél fejeződik be – a `return` utasítás elhagyható

Változók hatóköre

- Egy blokkban deklarált változók csak a deklarálástól kezdve a blokk végéig elérhetők
  - Következmény: az egyik metódusban deklarált `x` változó nem ugyanaz, mint a másik metódusbeli `x` változó
- A hívó környezet változói nem érhetők el a metódusban
  - Ezért szükséges a paraméter átadás és a visszatérési érték
  - Közös adat használható: globális változók, de használatuk nem javasolt

Egyszerű példa

Téglalap területének számítása

```
static int terület(int a, int b) //paraméterek átadása
{
    return a * b;
}

static void Main()
{
    int egyikOldal = 5;
    int másikOldal = 7;
    Console.WriteLine("A téglalap területe: "
        + terület(egyikOldal, másikOldal));
}
```

Tartalom

Metódusok paraméterei

- A paramétereknél megadandó a típus és az a név, amellyel a metódustörzsben a paraméterre hivatkozunk
- A paraméter átadásának kétféle módja van: érték szerinti és cím szerinti
- Érték szerinti: a paraméterként megadott változóról másolat készül, a metódusban ezzel a másolattal dolgozunk. Ha a metódusban megváltoztatjuk a paraméter értékét, az az eredeti változóra nincs hatással, a hívó környezetben nem érvényesül a módosítás.
- Cím szerinti: egy referenciát adunk át a paraméterként megadott változóra. A változóról nem készül másolat, tehát ha a metódusban megváltoztatjuk a paraméter értékét, akkor ez a módosítás a hívó környezetbeli változóra is érvényes lesz.

Metódusok paraméterei

- Az érték szerinti paraméter átadás az alapértelmezett
  - Referencia típusú változóknál az érték szerinti paraméter átadás azt jelenti, hogy a változóban tárolt cím másolódik le és adódik át, azaz ha a metódusban módosítjuk a referencia által hivatkozott objektumot, annak a hatása kívül is látszik
- Ha érték típusú paramétert szeretnénk cím szerint átadni, azt a `ref` vagy az `out` kulcsszó használatával tehetjük meg
- Mivel két fajta változótípus van, az érték- és a referencia típus, valójában 4 különféle paraméterátadási módunk van

Érték típusok érték szerinti paraméter átadása

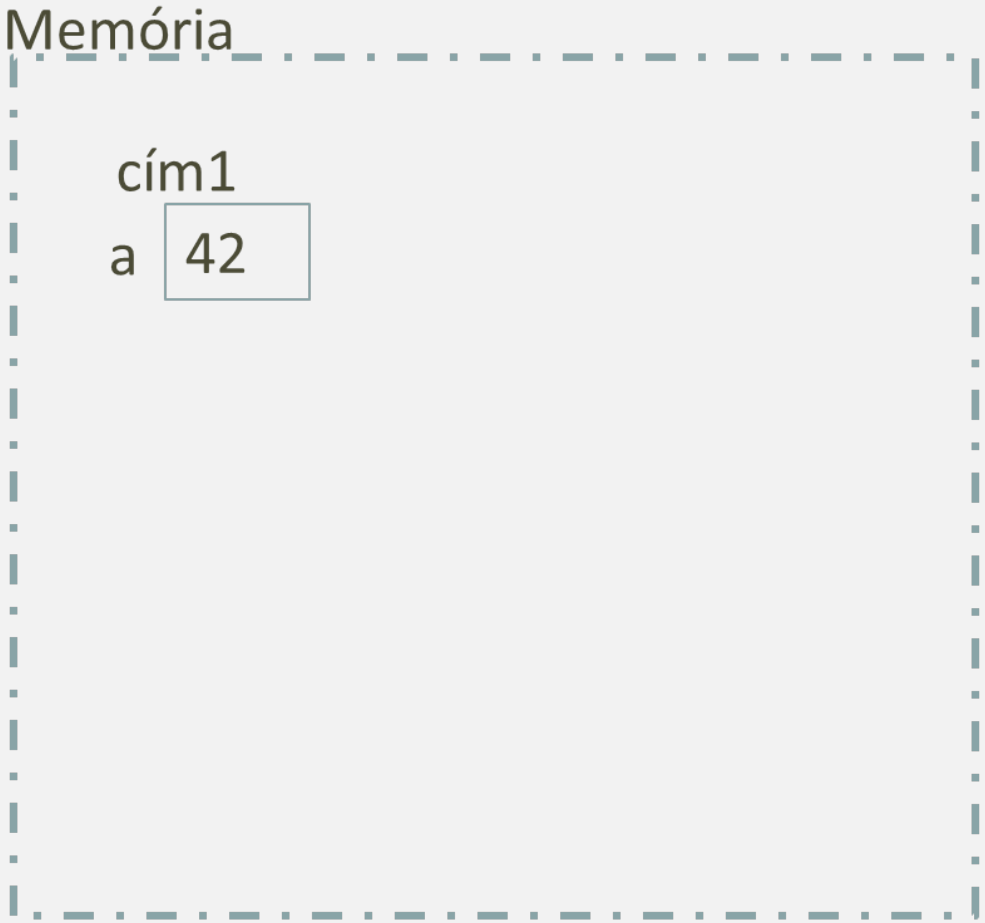
```
static void novel(int bemenet)
{
    bemenet++;
}

static void csere(int elso, int masodik)
{
    int temp = elso;
    elso = masodik;
    masodik = temp;
}
```

Érték típusok érték szerinti paraméter átadása

```
static void Main()
{
    int a = 42, b = 23;
    novel(a);
    csere(a, b);
}
```

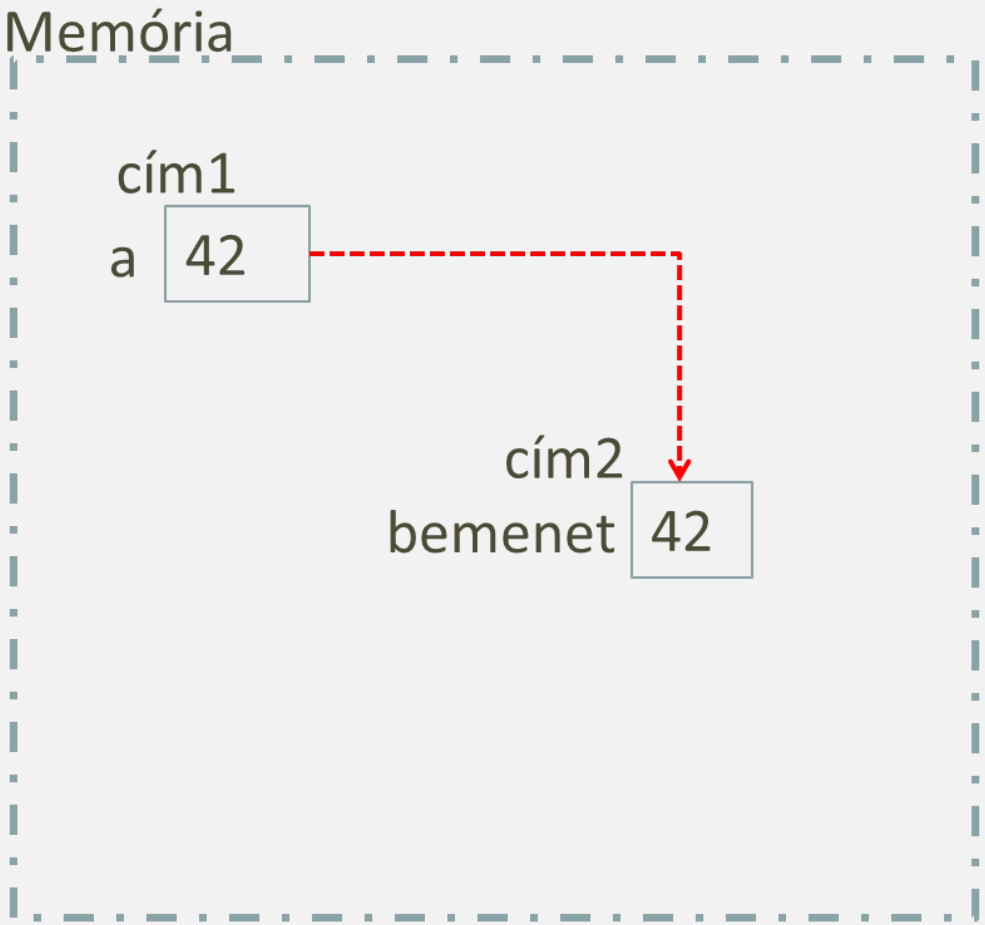
- A híváskor a két változó értékéről másolat készül, és a másolatot adjuk át paraméterként a hívott metódusoknak
- A hívott metódusokban a paraméterek módosítása semmilyen hatással nincs ezen változók értékére

Érték típusok érték szerinti paraméter átadása



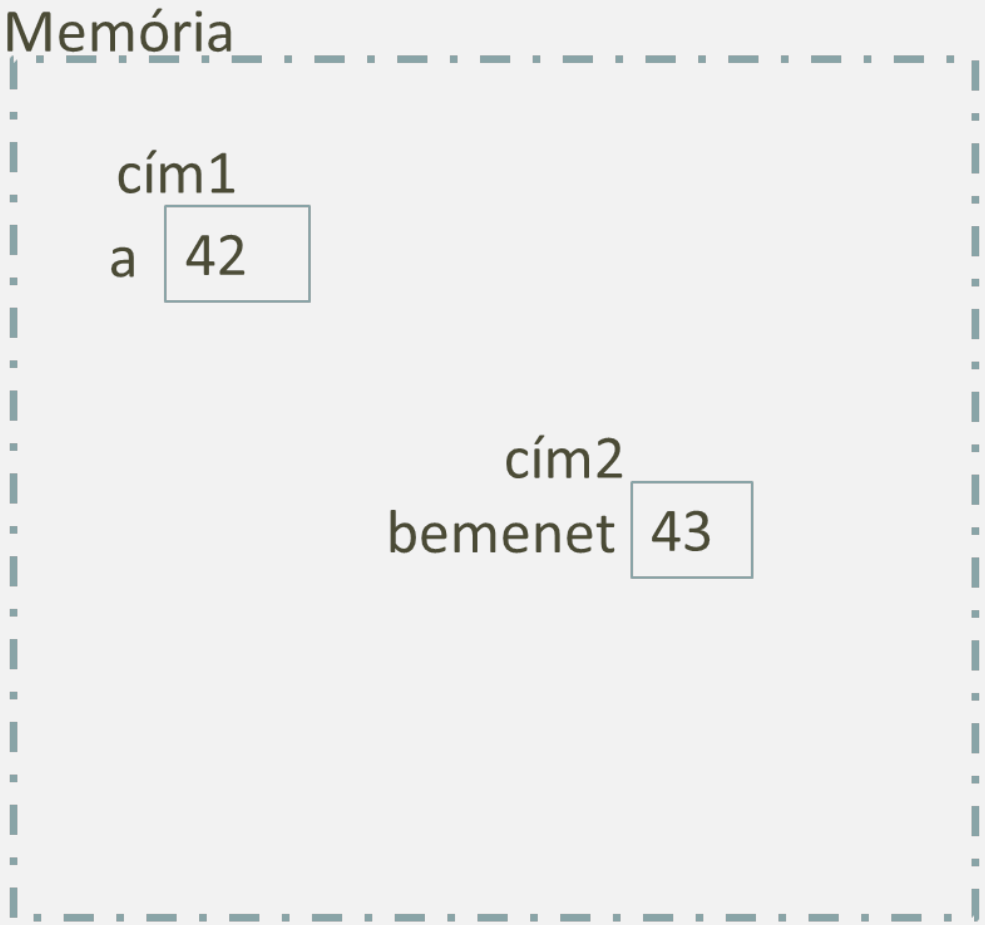
int a=42;

Érték típusok érték szerinti paraméter átadása



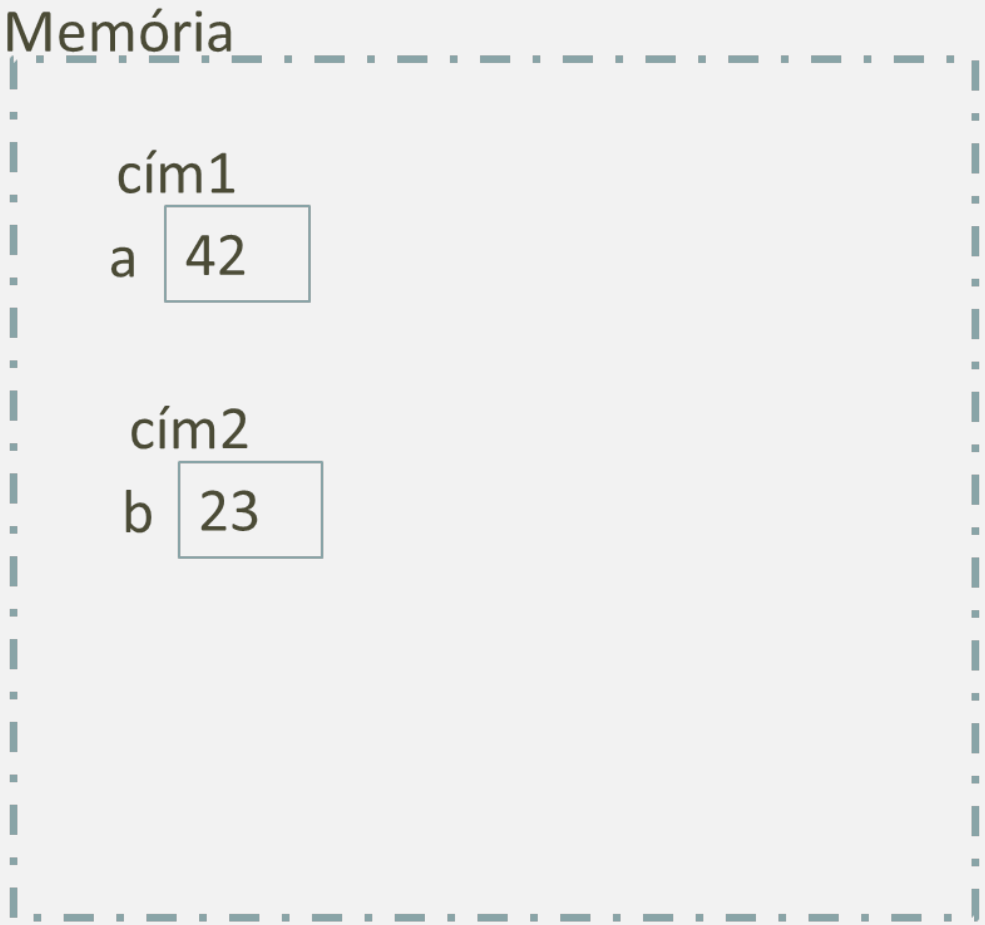

```
int a=42;
novel(a);
```

Érték típusok érték szerinti paraméter átadása



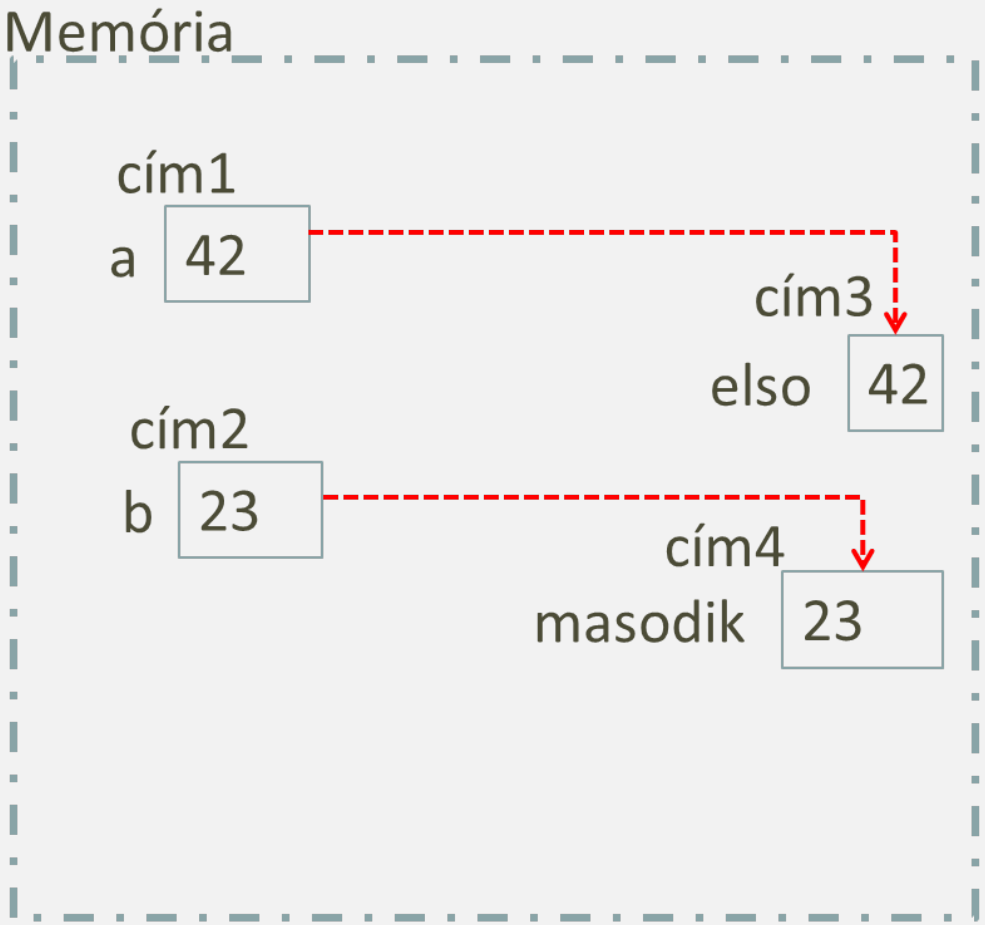

```
int a=42;
novel(a);
        bemenet++;
```

Érték típusok érték szerinti paraméter átadása



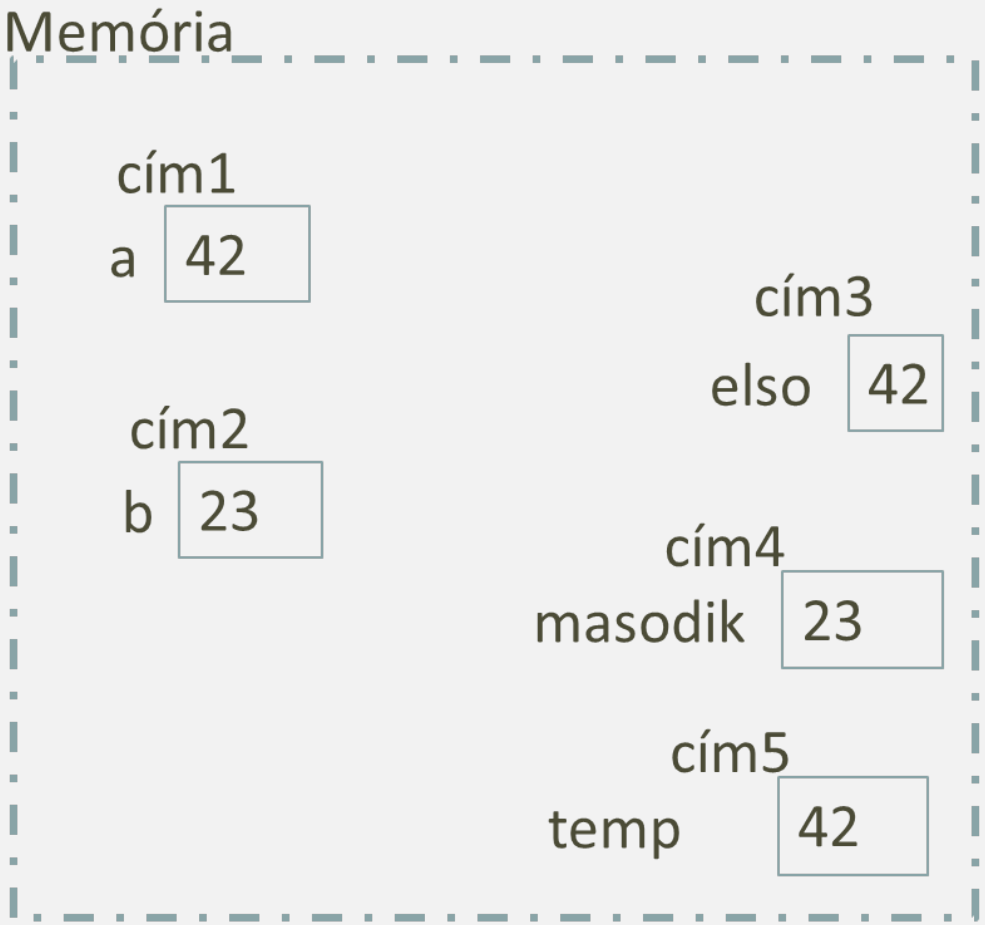

```
int a=42, b=23;
```

Érték típusok érték szerinti paraméter átadása



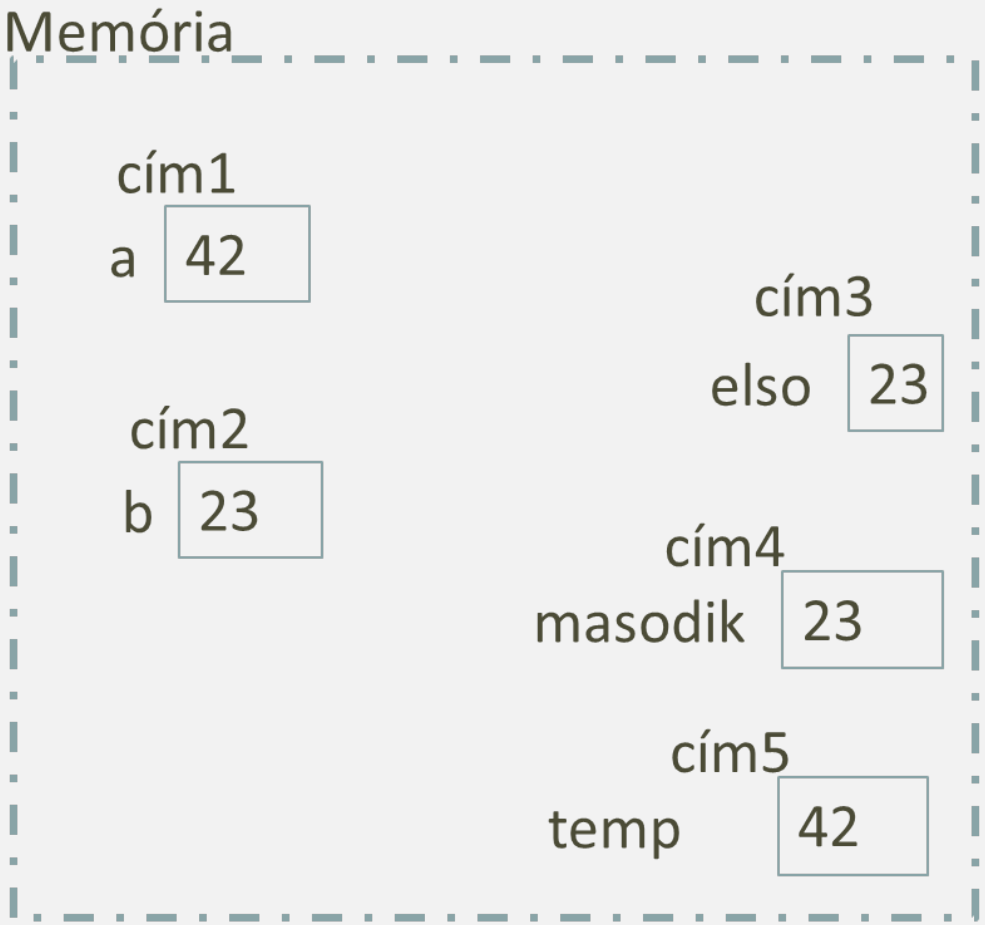

```
int a=42, b=23;
csere(a,b);
```

Érték típusok érték szerinti paraméter átadása



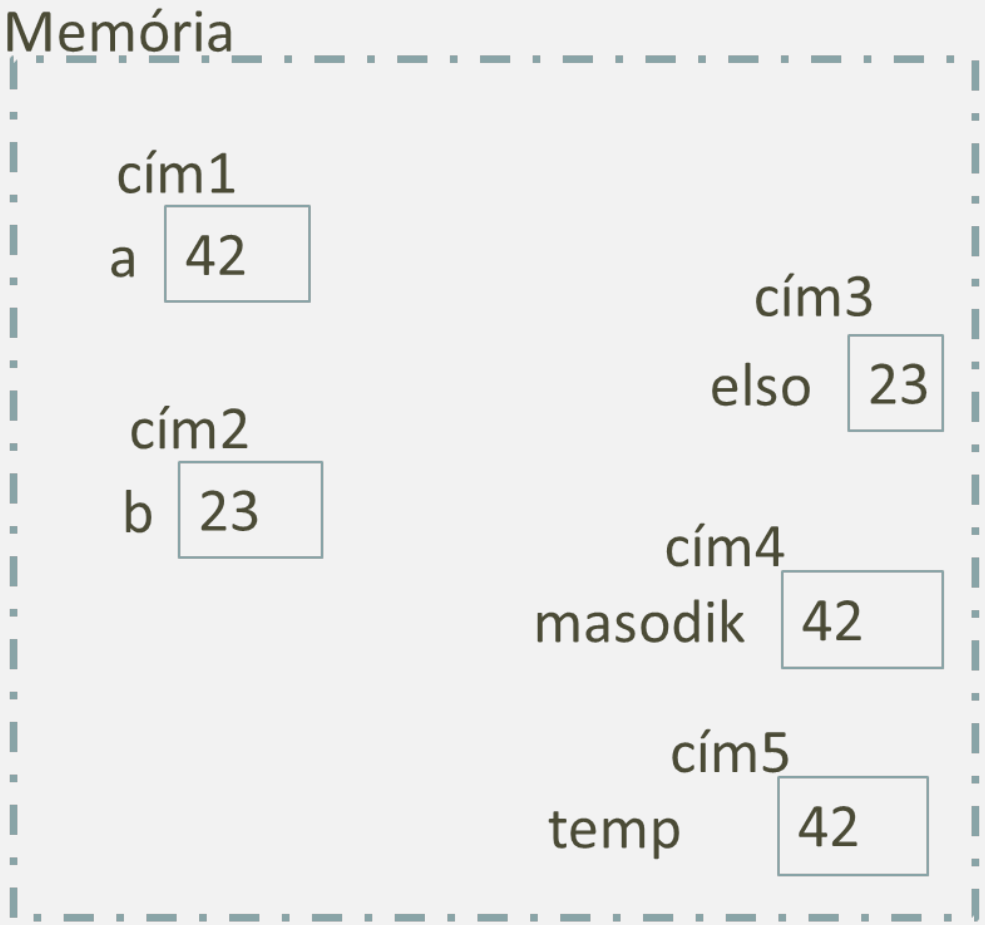

```
int a=42, b=23;
csere(a,b);
        int temp;
        temp=elso;
```

Érték típusok érték szerinti paraméter átadása




```
int a=42, b=23;
csere(a,b);
        int temp;
        temp=elso;
        elso=masodik;
```

Érték típusok érték szerinti paraméter átadása




```
int a=42, b=23;
csere(a,b);
        int temp;
        temp=elso;
        elso=masodik;
        masodik=temp;
```

Érték típusok cím szerinti paraméter átadása

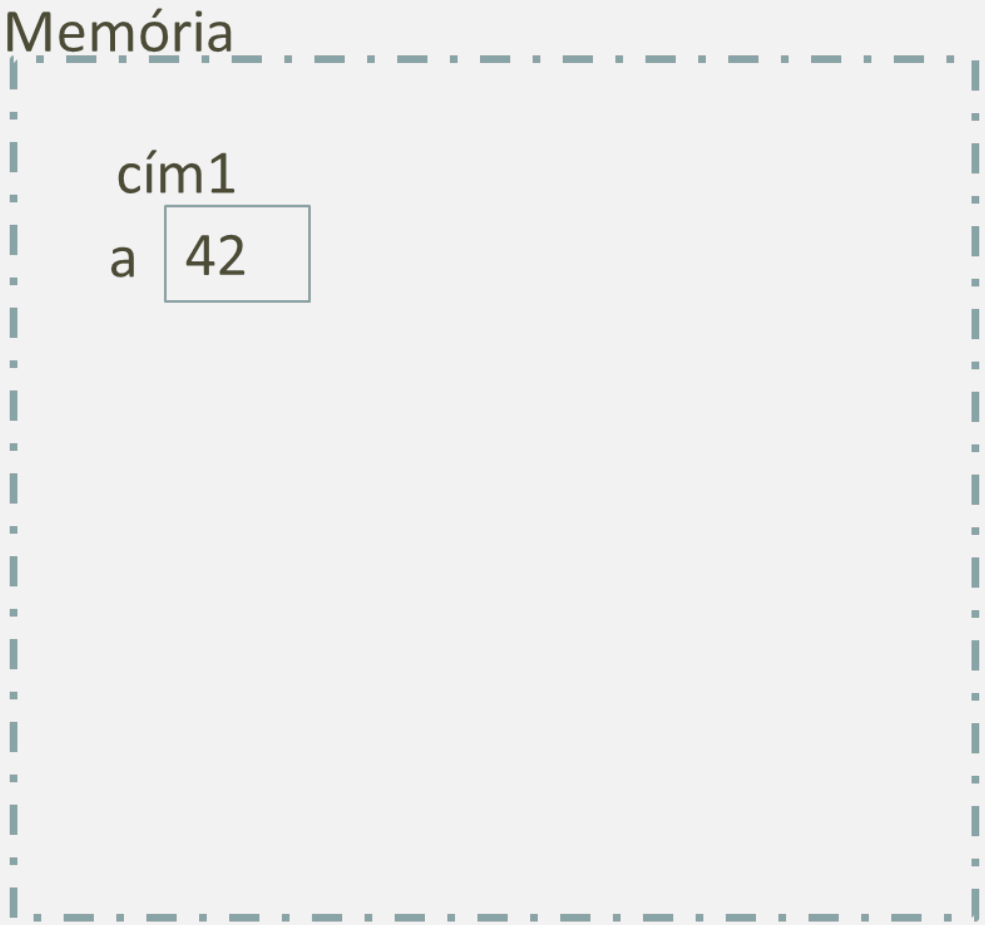
```
static void novel(ref int bemenet)
{
    bemenet++;
}

static void csere(ref int elso, ref int masodik)
{
    int temp = elso;
    elso = masodik;
    masodik = temp;
}
```

Érték típusok cím szerinti paraméter átadása

```
static void Main()
{
    int a = 42, b = 23;
    novel(ref a);
    csere(ref a, ref b);
}
```

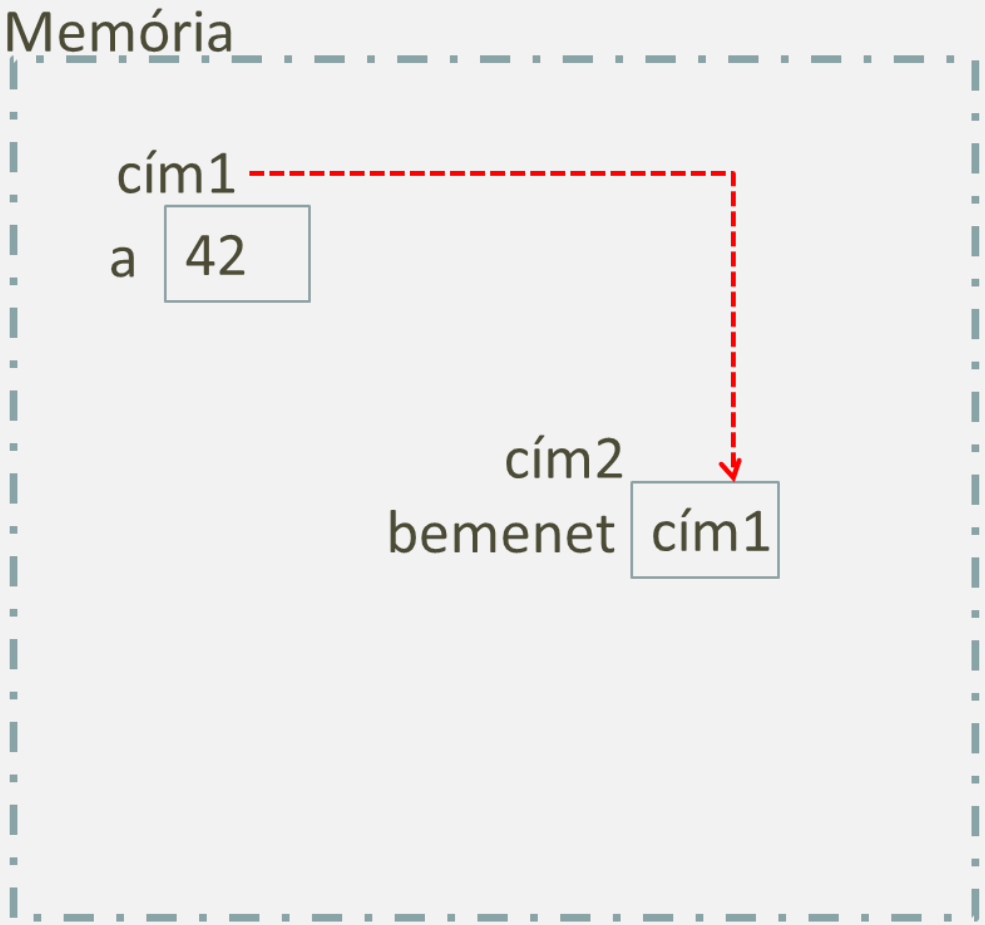
- A híváskor a változók címéről másolat képződik, és a címek kerülnek át paraméterként a hívott metódusokba
- A hívott metódusokban a paraméterek módosítása gyakorlatilag ezen változók módosítását jelenti

Érték típusok cím szerinti paraméter átadása



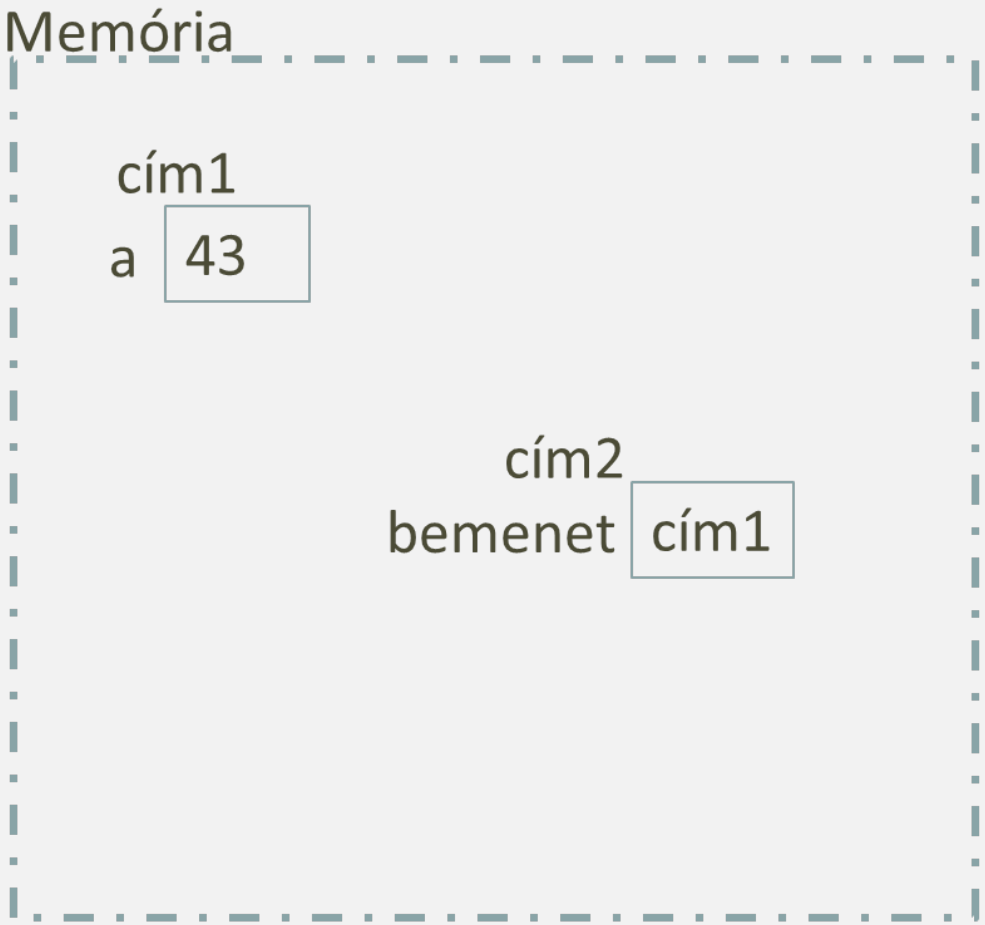
int a=42;

Érték típusok cím szerinti paraméter átadása



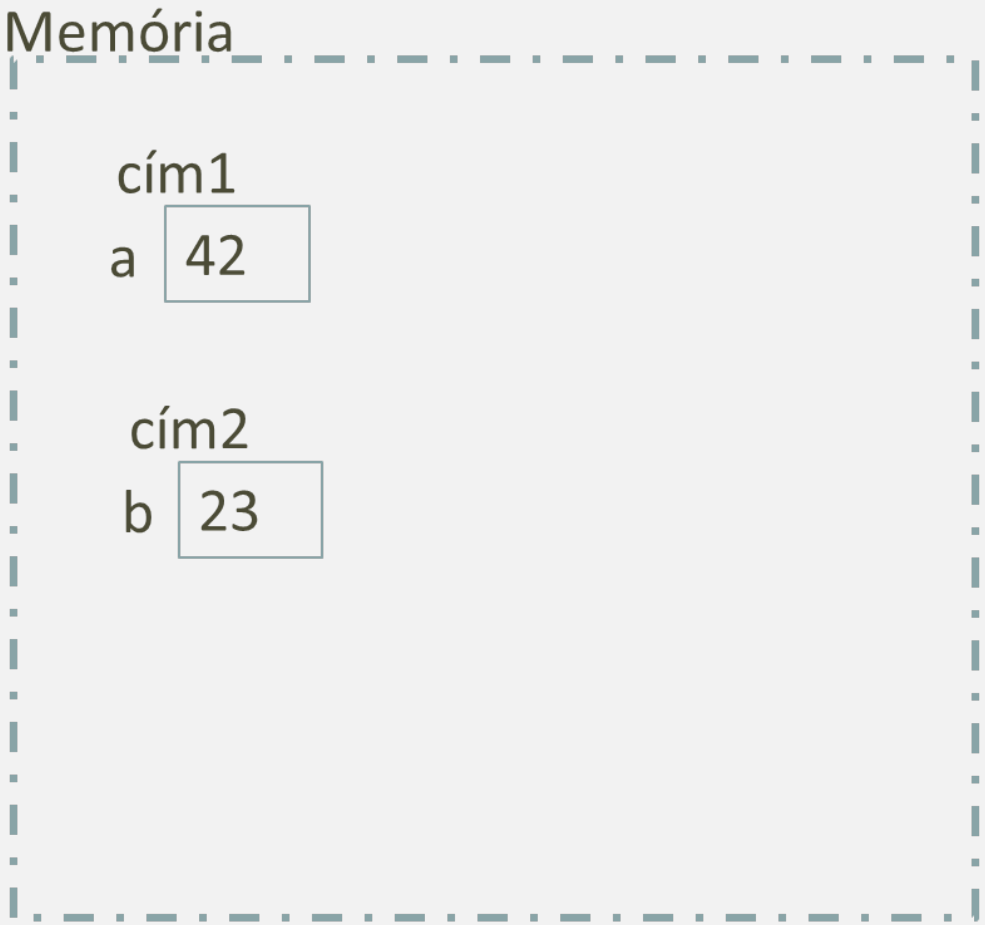

```
int a=42;
novel(ref a);
```

Érték típusok cím szerinti paraméter átadása



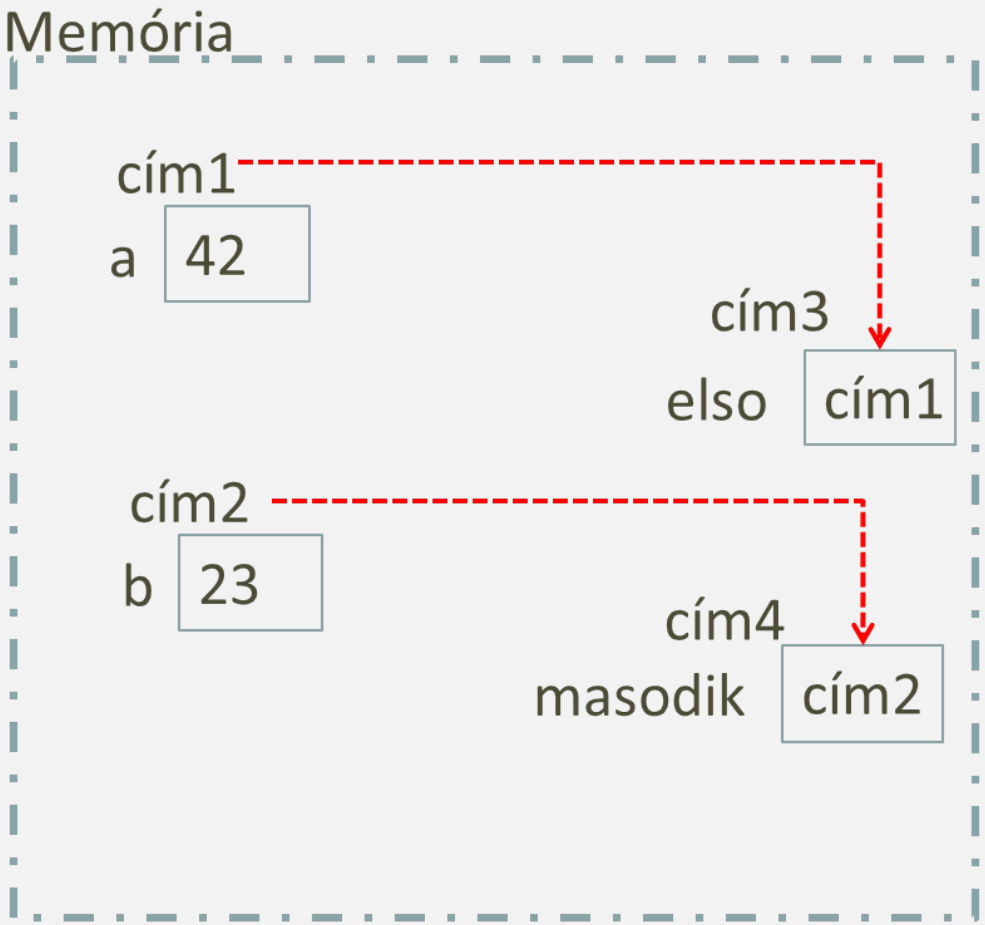

```
int a=42;
novel(ref a);
        bemenet++;
```

Érték típusok cím szerinti paraméter átadása



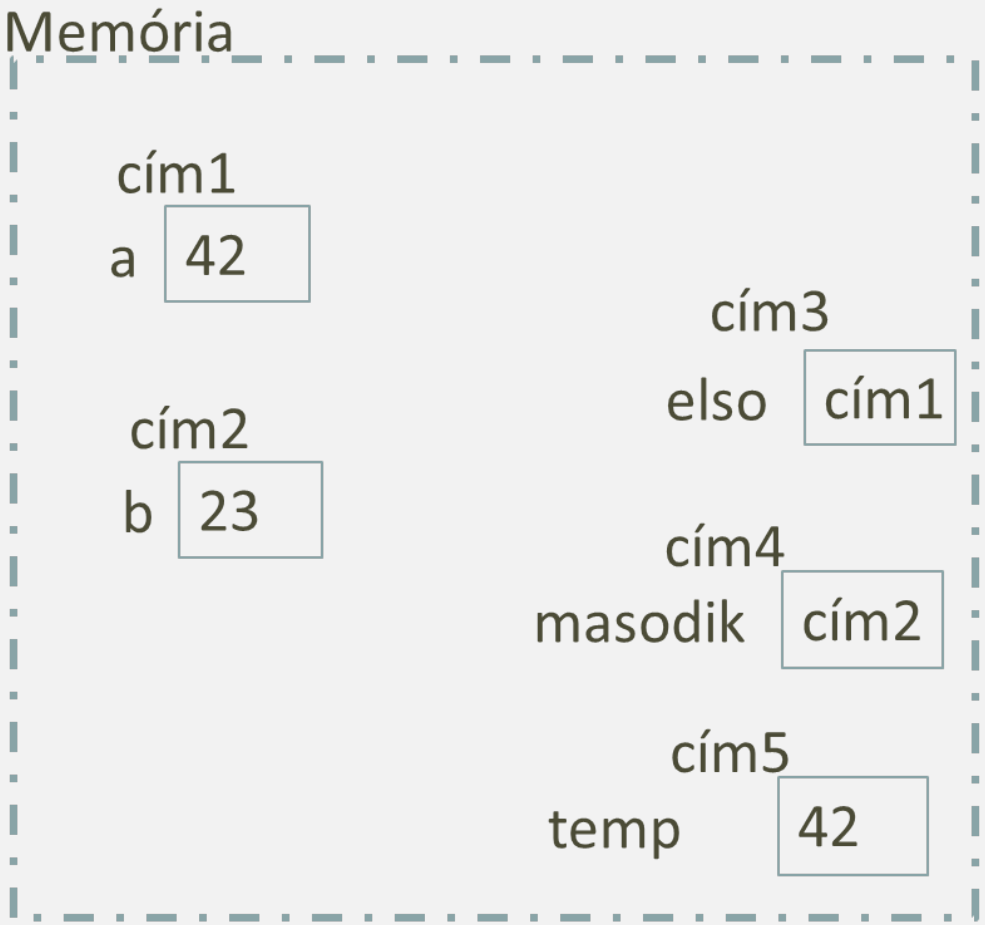

```
int a=42, b=23;
```

Érték típusok cím szerinti paraméter átadása



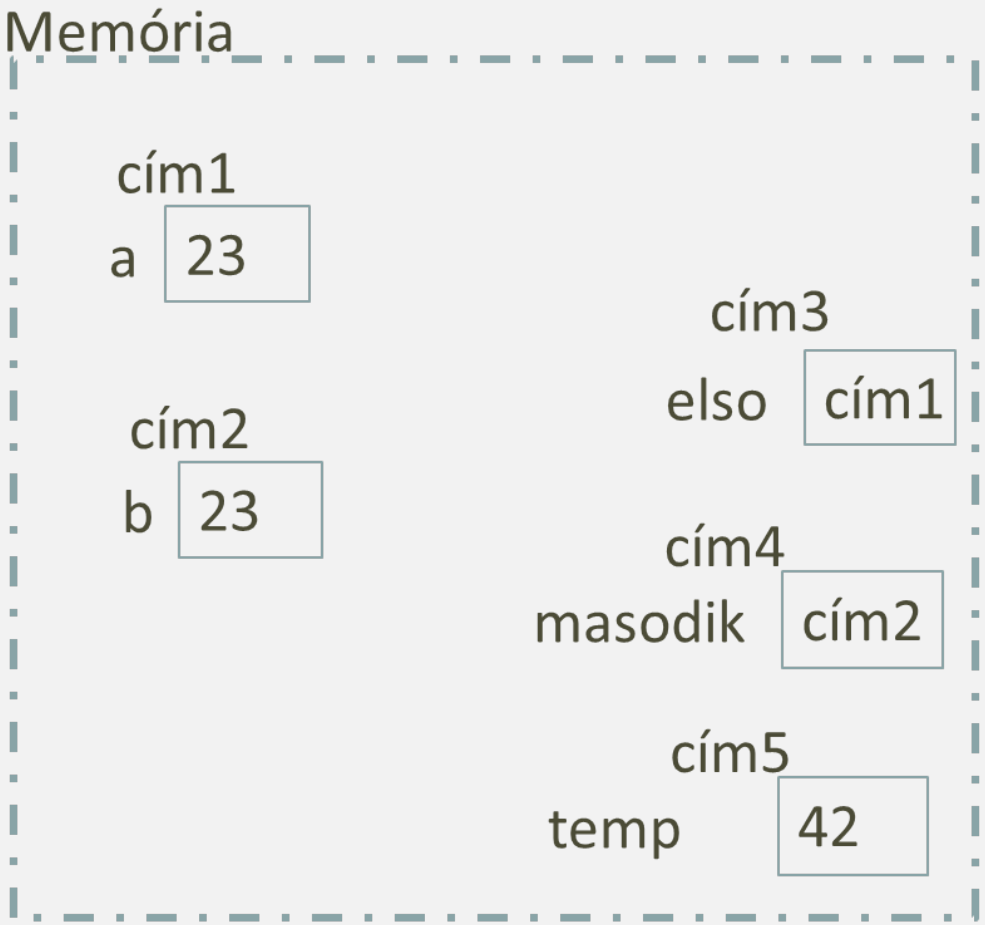

```
int a=42, b=23;
csere(ref a, ref b);
```

Érték típusok cím szerinti paraméter átadása



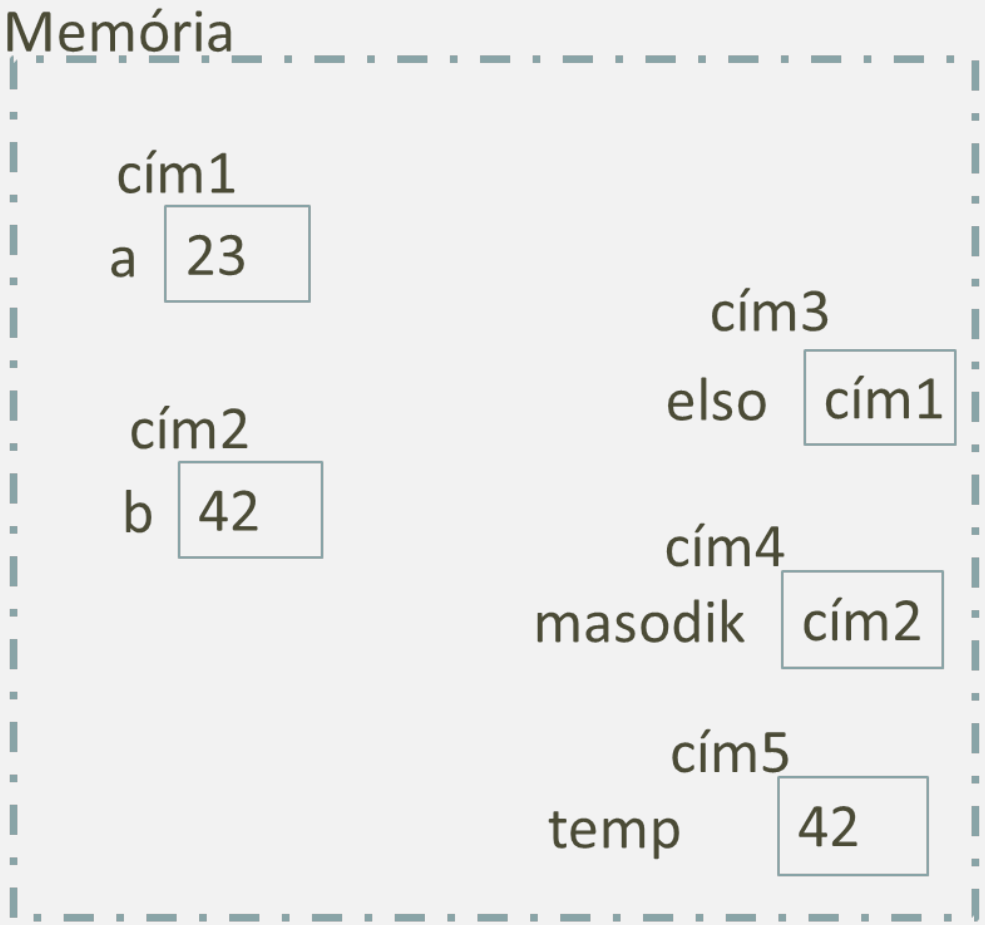

```
int a=42, b=23;
csere(ref a, ref b);
        int temp;
        temp=elso;
```

Érték típusok cím szerinti paraméter átadása




```
int a=42, b=23;
csere(ref a, ref b);
        int temp;
        temp=elso;
        elso=masodik;
```

Érték típusok cím szerinti paraméter átadása




```
int a=42, b=23;
csere(ref a, ref b);
        int temp;
        temp=elso;
        elso=masodik;
        masodik=temp;
```

Referencia típusok érték szerinti paraméter átadása

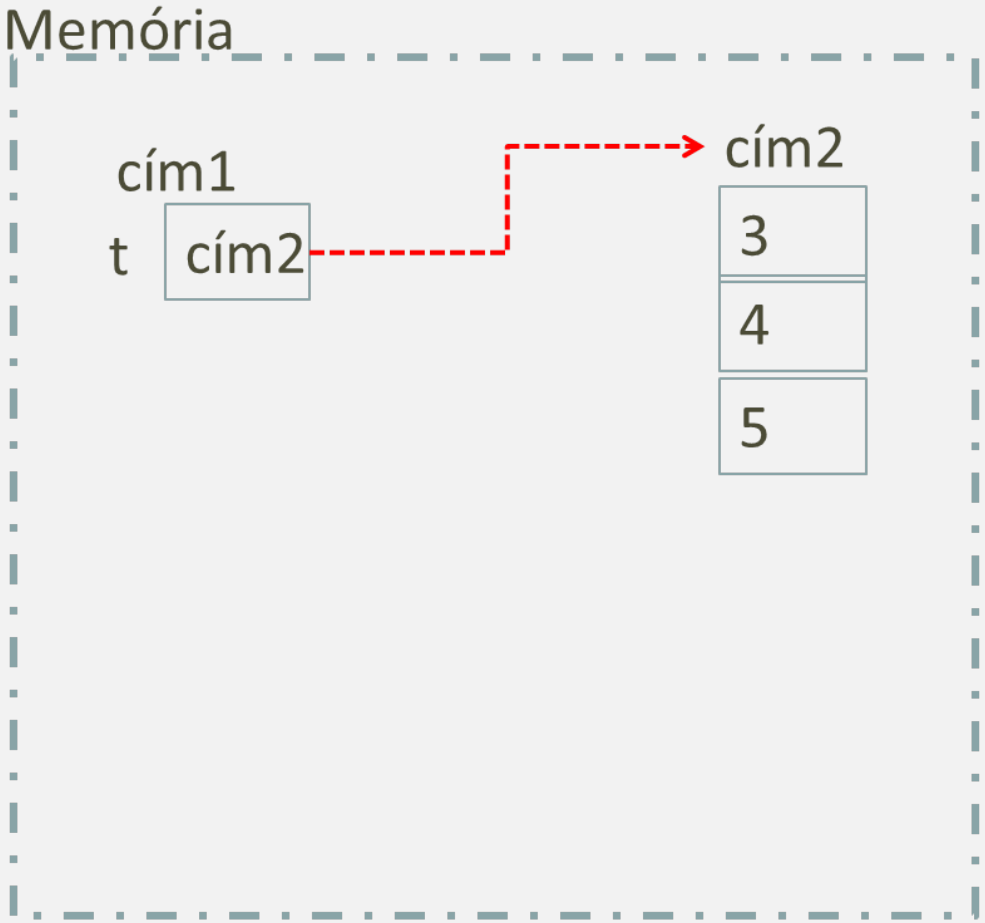
```
static void tombkez1(int[] arr)
{
    arr[1] = 42;
}

static void tombkez2(int[] arr)
{
    arr = new int[3] {23, 23, 23};
}
```

Referencia típusok érték szerinti paraméter átadása

```
static void Main()
{
    int[] t = {3, 4, 5};
    tombkez1(t);
    tombkez2(t);
}
```

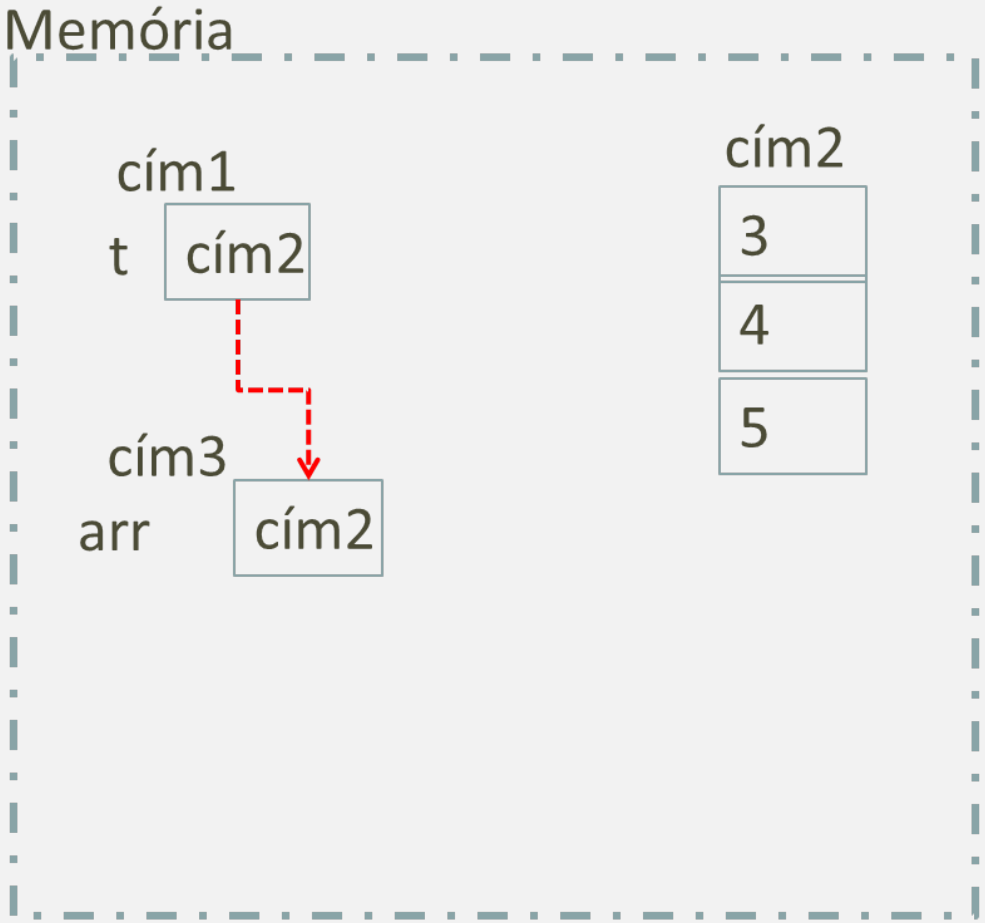
- A metódusok hívásakor a változók értékéről készül másolat, és az érték kerül át paraméterként a hívott eljárásba → az átadott érték viszont most egy referencia
- A hívott metódusokban nem lehet módosítani ezt a referenciát → a hivatkozott memóriaterületen lévő értékeket viszont lehet

Referencia típusok érték szerinti paraméter átadása



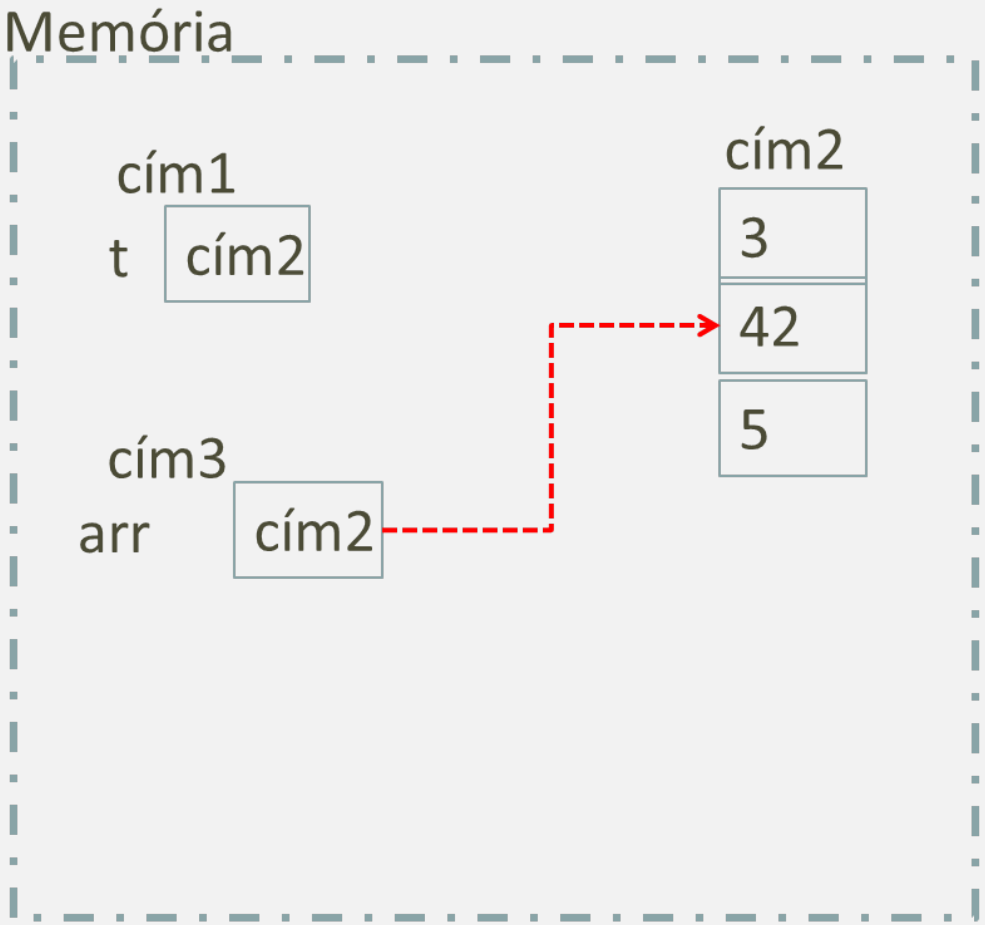
int[] t= {3, 4, 5};

Referencia típusok érték szerinti paraméter átadása



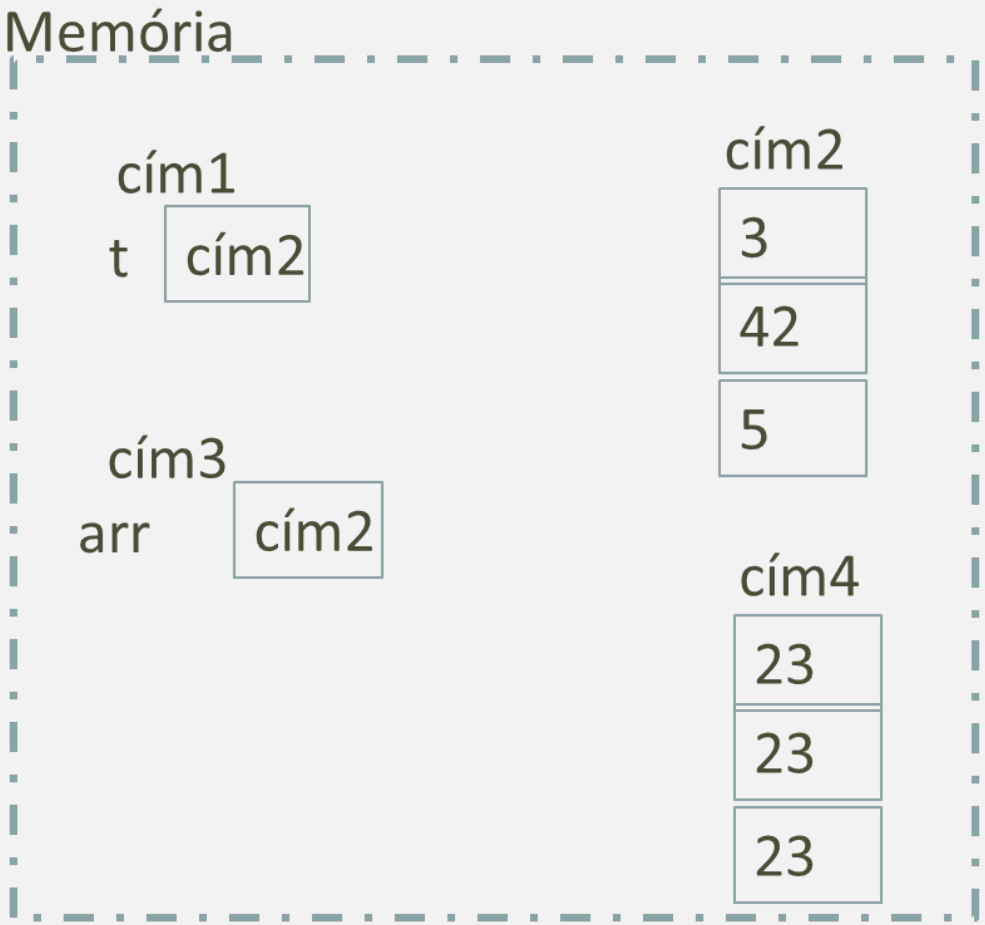

```
int[] t= {3, 4, 5};
tombkez1(t);
```

Referencia típusok érték szerinti paraméter átadása



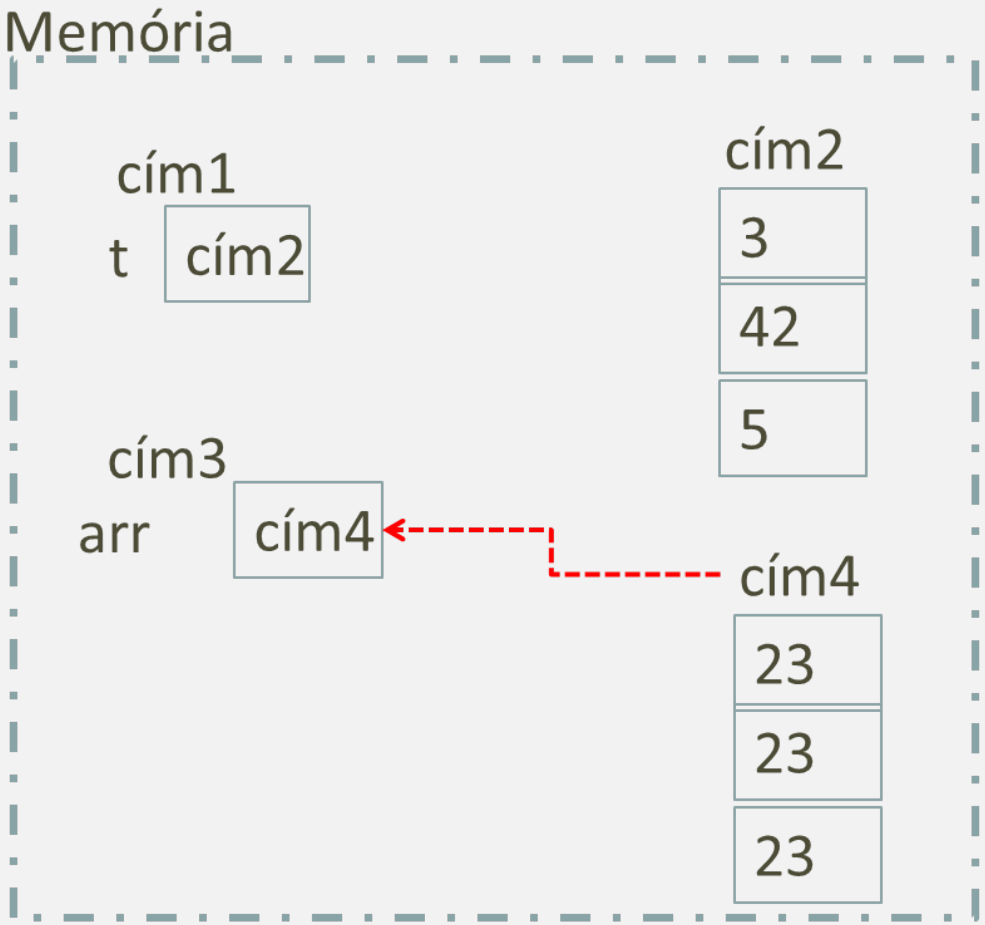

```
int[] t= {3, 4, 5};
tombkez1(t);
        arr[1]=42;
```

Referencia típusok érték szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(t);
        arr[1]=42;
tombkez2(t);
        arr=new int[3]
        { 23, 23, 23};
```

Referencia típusok érték szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(t);
        arr[1]=42;
tombkez2(t);
        arr=new int[3]
        { 23, 23, 23};
```

Referencia típusok cím szerinti paraméter átadása

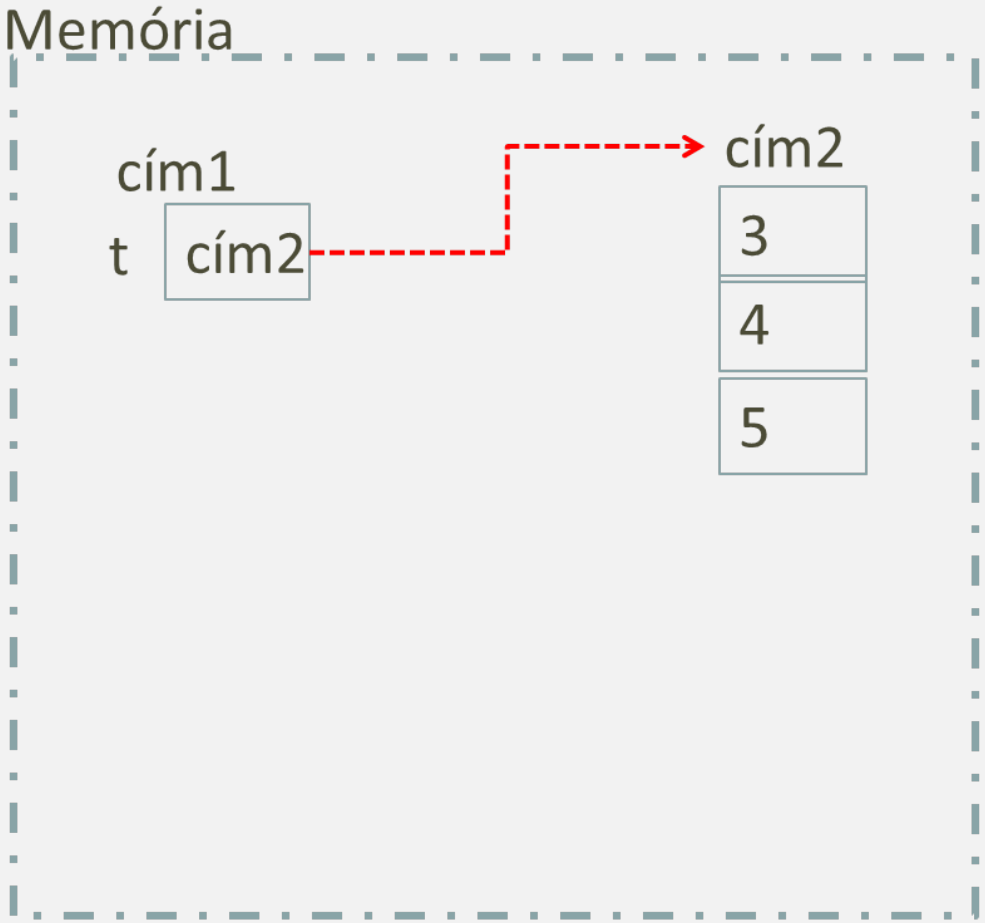
```
static void tombkez1(ref int[] arr)
{
    arr[1] = 42;
}

static void tombkez2(ref int[] arr)
{
    arr = new int[3] {23, 23, 23};
}
```

Referencia típusok cím szerinti paraméter átadása

```
static void Main()
{
    int[] t = {3, 4, 5};
    tombkez1(ref t);
    tombkez2(ref t);
}
```

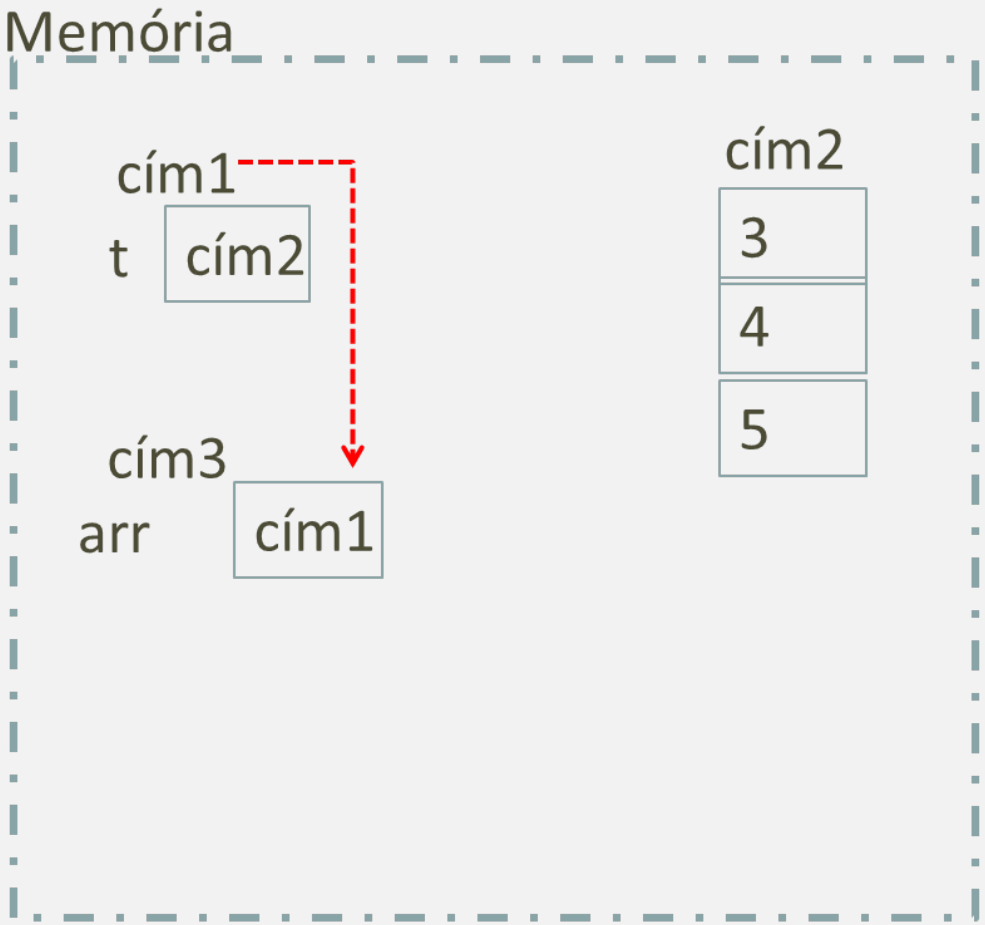
- A metódusok hívásakor a változókról címéről másolat készül, és a cím kerül át paraméterként a hívott metódusokba → Referenciára mutató referencia
- A hívott metódusokban ezt a változót és a hivatkozott memóriaterületet is lehet módosítani

Referencia típusok cím szerinti paraméter átadása



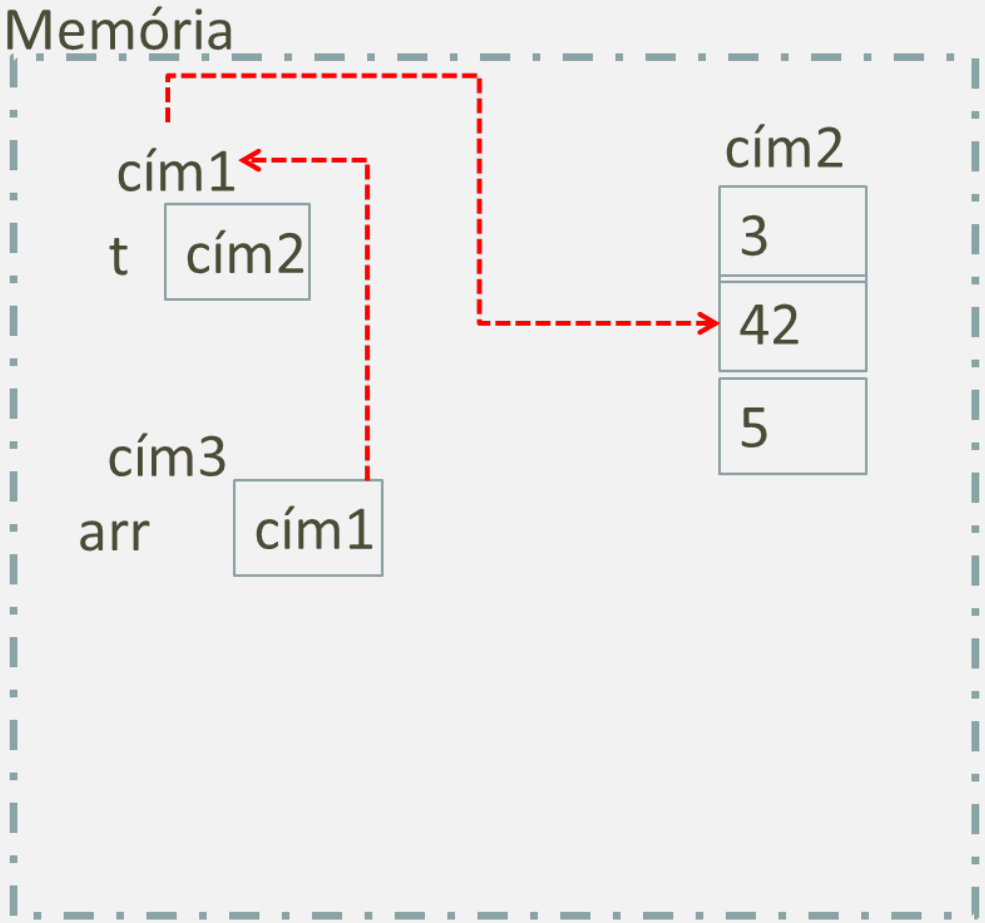
int[] t= {3, 4, 5};

Referencia típusok cím szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(ref t);
```

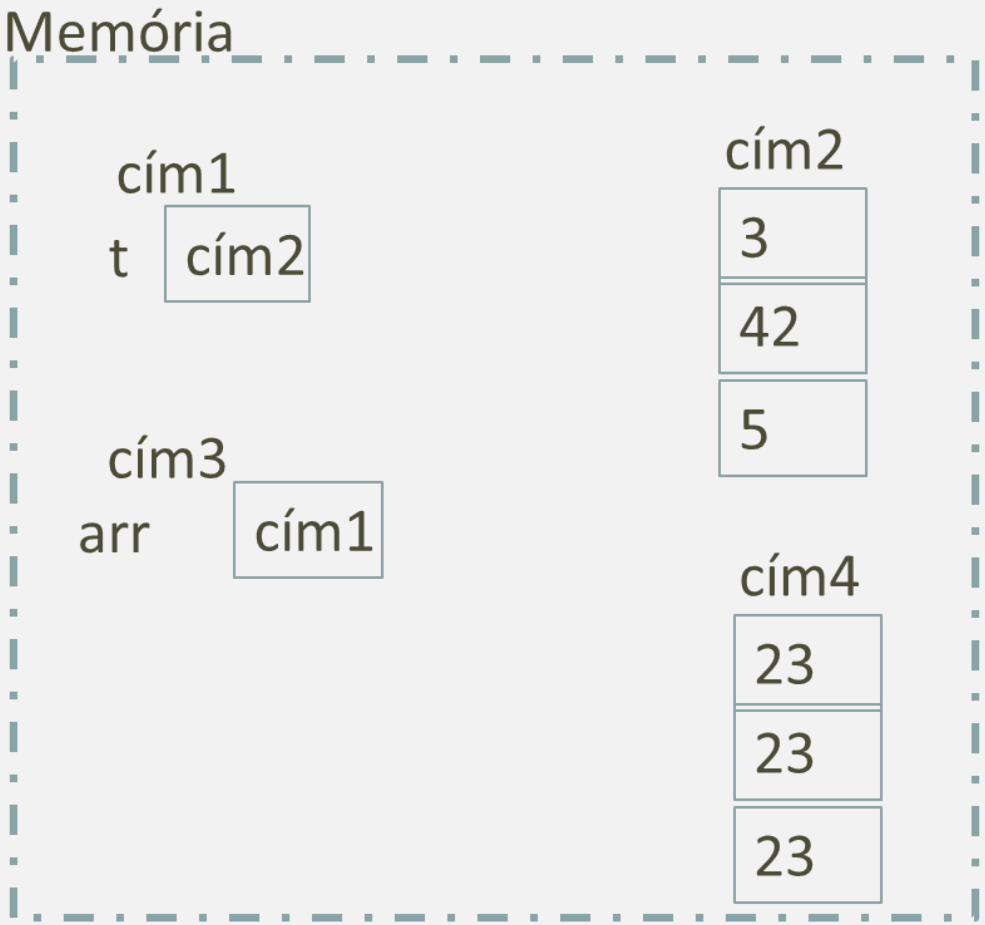
Referencia típusok cím szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(ref t);
        arr[1]=42;
```

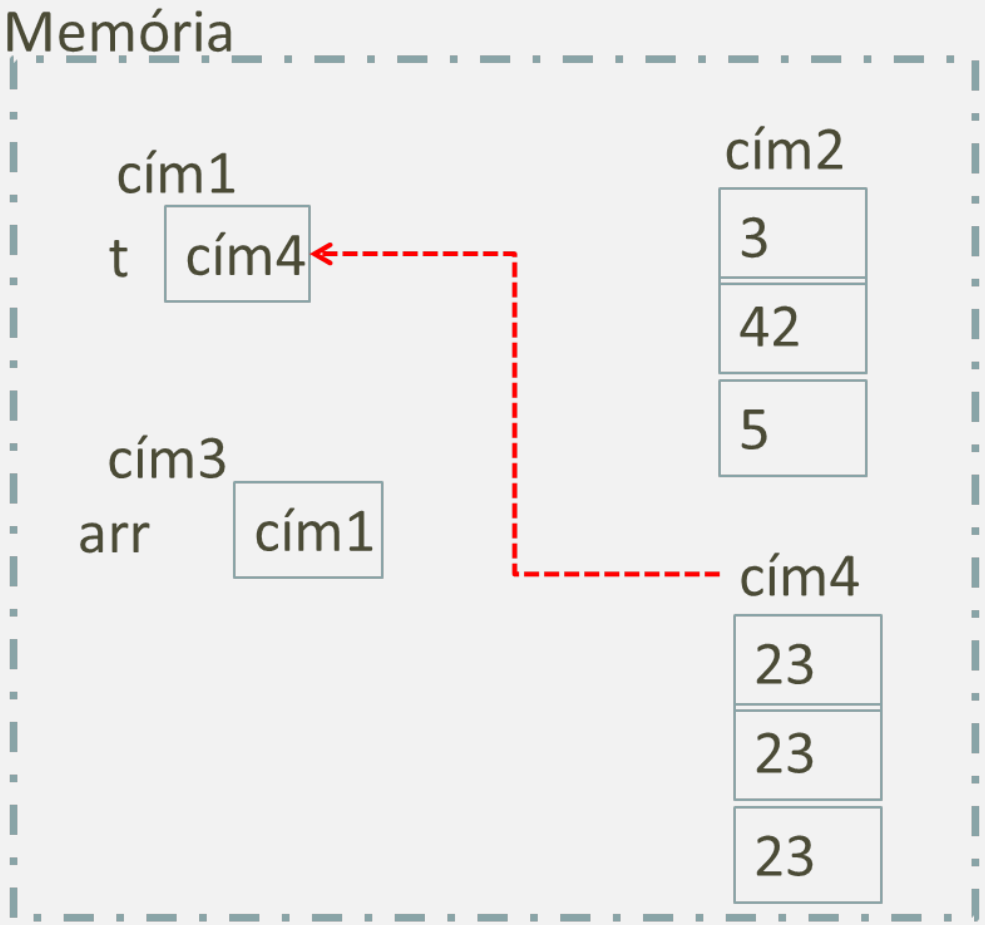
Teljesen mindegy, hogy hány referencián keresztül van az értékadás, mindig a megfelelő értékre történik a hivatkozás!

Referencia típusok cím szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(ref t);
        arr[1]=42;
tombkez2(ref t);
        arr=new int[3]
        { 23, 23, 23};
```

Referencia típusok cím szerinti paraméter átadása




```
int[] t= {3, 4, 5};
tombkez1(ref t);
        arr[1]=42;
tombkez2(ref t);
        arr=new int[3]
        { 23, 23, 23};
```

out típusú paraméter átadása

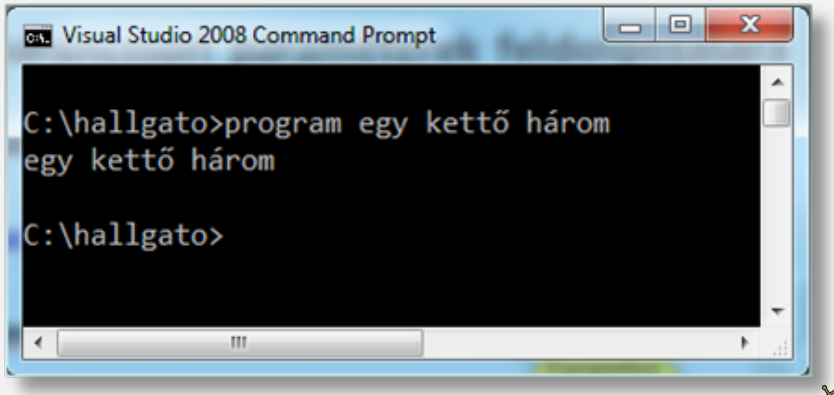
- Az out módosító kulcsszó használata hasonló a `ref` használatához. A paraméter ilyenkor is cím szerint adódik át
- Különbség a `ref`-hez képest, hogy az átadott változót nem kell inicializálni a használat előtt

```
static void beker(out int szam)
{
Console.Write("Kérek egy számot: ");
szam = int.Parse(Console.ReadLine());
}

static void Main()
{
    int x;
    beker(out x);
    Console.WriteLine("A megadott szam: " + x);
}
```

Parancssori paraméterek feldolgozása

A `Main()` metódusnak is lehetnek paraméterei

```
static void Main(string[] args)
{
    for (int i = 0; i < args.Length; i++)
        Console.Write(args[i] + " ");
}
```



Metódusnév túlterhelés

- Metódusnév túlterhelés (overloading) – több azonos nevű metódus (egy osztályon belül)
- A metódusok szignatúrája egyedi az osztályon belül
  - Szignatúra = metódusnév + paraméterek száma, típus és sorrendje
- Használhatunk több azonos nevű metódust is (egy osztályon belül), de a paraméterlistájuknak különbözni kell

Metódusnév túlterhelés

Különböző metódusok azonos névvel

```
void F() { ... }
void F(int x) { ... }
void F(ref int x) { ... }
void F(int x, int y) { ... }
int F(string s) { ... }
void F(string[] a) { ... }
```

Hibás túlterhelés

```
int F(int x) { ... }
```

A paraméterekben nincs eltérés, csak a metódus típusában

```
void F(out int x) { ... }
```

`ref` helyett `out`: ugyanúgy cím szerinti a paraméter átadás

Felhasznált irodalom

- Szlávi Péter, Zsakó László: Módszeres programozás: Programozási tételek (Mikrológia 19). ELTE TTK, 2002
- Korábbi félévek OOP diasorai