Objektumorientált Programozás V.

A Microsoft Visual Studio 2010 használata
Műveletek tömbökkel
Érték- és referenciatípusú változók
Feladatok

Hallgatói Tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Objektumorientált Programozás V.

A Microsoft Visual Studio 2010 használata

Műveletek tömbökkel

Érték- és referenciatípusú változók

Feladatok

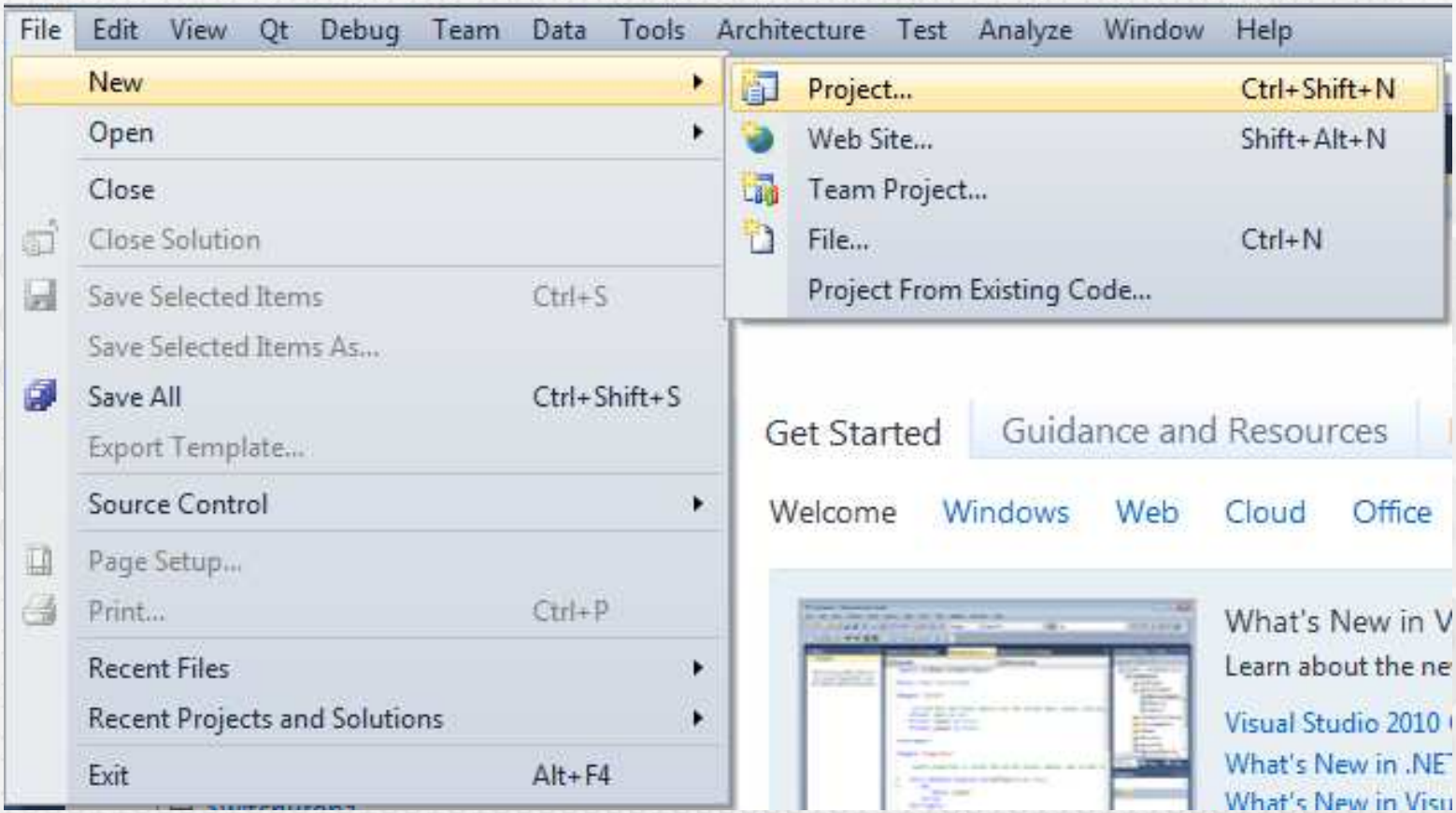
Projektek és megoldások

- **Projekt („Project”)**
  A projekt egy futtatható programhoz vagy más típusú szoftvermodulhoz tartozó, együtt kezelt szoftverelemek (többségében fájlok) összessége.
- **C# forráskód („source code”) [*.cs]**
- **Hivatkozások („references”)**
- **Beállítások („settings”) [*.settings]**
- **Konfigurációs fájlok („configuration”) [*.config]**
- **Egyéb erőforrások („resources”) [*.resx, *.rc, *.resources]**
  A projekthez tartozó elemek mappák létrehozásával hierarchikus fastruktúrába rendezhetők.
  A C# projekteket a Visual Studio *.csproj kiterjesztésű fájlokban tárolja.

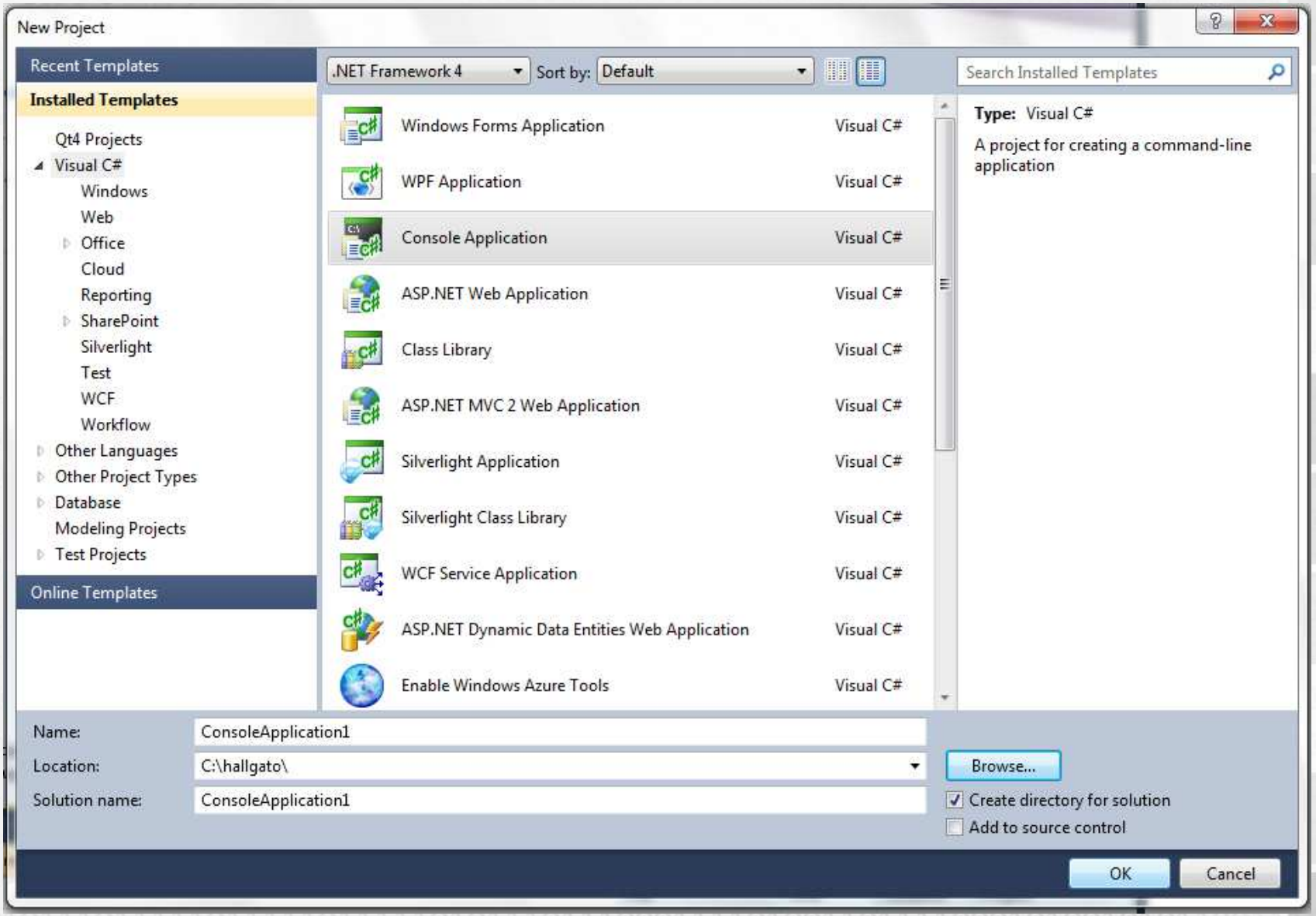
Projektek és megoldások

- **Megoldás („Solution”)**
  A megoldás több összefüggő projekt együttes kezelését teszi lehetővé.
  Ezek a projektek virtuális mappák segítségével hierarchikus fastruktúrába is rendezhetők.
  A megoldásokat a Visual Studio *.sln kiterjesztésű fájlokban tárolja.

Új projekt létrehozása



Új projekt létrehozása



A legfontosabb projekttípusok

- **Grafikus Windows alkalmazás („Windows Forms Application”)**
  Végeredménye egy „exe” kiterjesztésű futtatható program.
- **Parancsértelmezőben futó Windows alkalmazás („Console Application”)**
  Végeredménye egy „exe” kiterjesztésű futtatható program.
- **Osztálykönyvtár („Class Library”)**
  Végeredménye egy „dll” kiterjesztésű könyvtárfájl.
- **Üres projekt („Empty Project”)**
  Ehhez a projekttípushoz kézzel kell a megfelelő elemeket hozzáadni.

Kód készítés

Program.cs*

ConsoleApplication1.Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Kód készítés

```
static void Main(string[] args)
{
    Console.
}
```

SetWindowSize
Title
TreatControlCAsInput
WindowHeight
WindowLeft
WindowTop
WindowWidth
Write
WriteLine

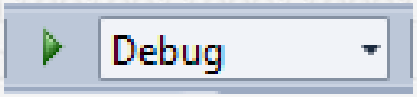
void Console.WriteLine(string format, params object[] arg) (+ 18 overload(s))
Writes the text representation of the specified array of objects, followed by the current line terminator, to the standard output stream using the specified format information.

Exceptions:
System.IO.IOException
System.ArgumentNullException
System.FormatException

A program

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello C# World!");
            Console.ReadLine();
        }
    }
}
```

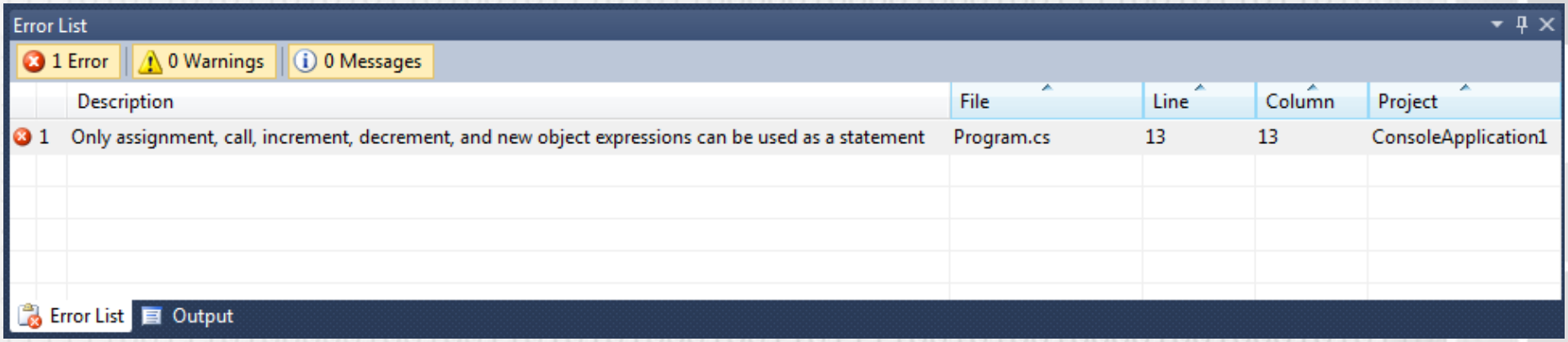
Futtatás


Debug

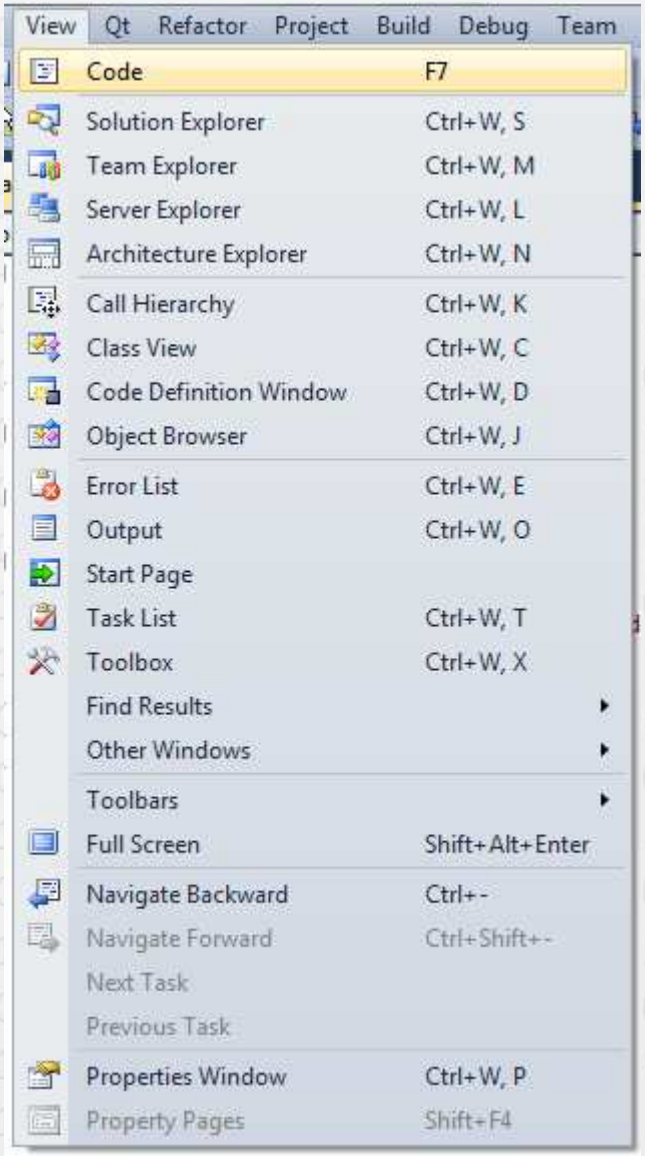

Build
Debug
Team
Data
Tools
Architecture
Test
Analyz
Build Solution
F6
Rebuild Solution
Clean Solution
Build ConsoleApplication1
Shift+F6
Rebuild ConsoleApplication1
Clean ConsoleApplication1
Publish ConsoleApplication1
Run Code Analysis on ConsoleApplication1
Batch Build...
Configuration Manager...

Hibás program

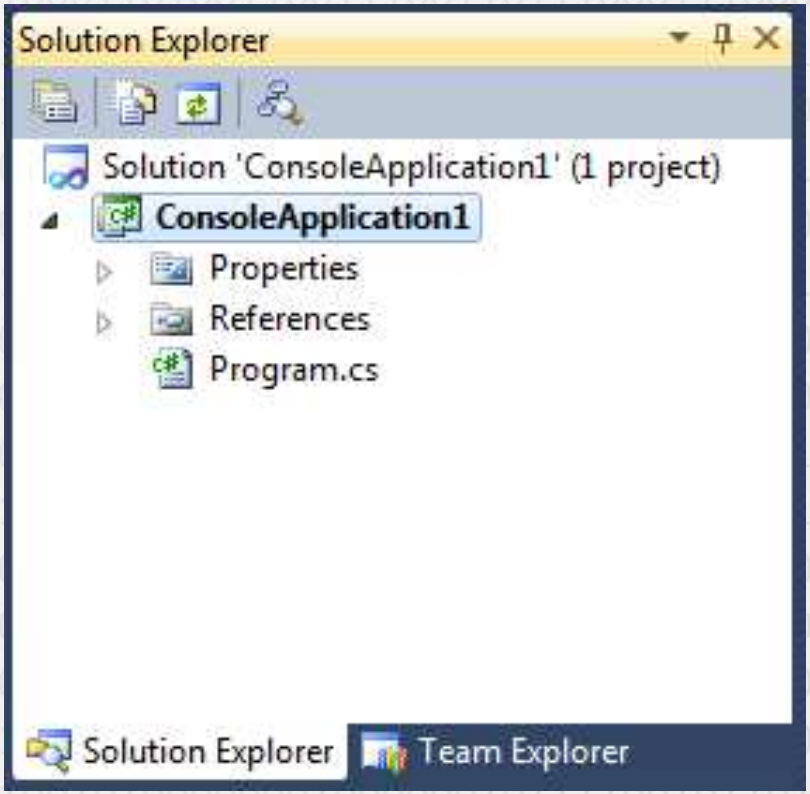
```
static void Main(string[] args)
{
    Console.WriteLine("Hello C# World!");
    Console.ReadLine;
}
```



A View menüpont

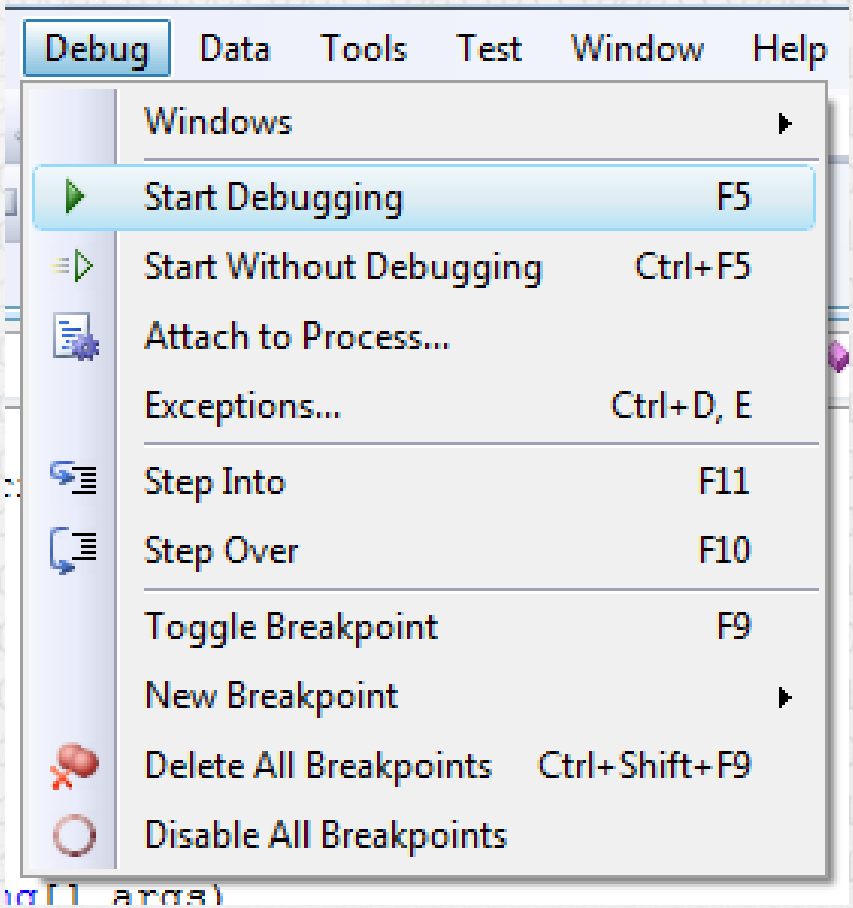


A Solution Explorer



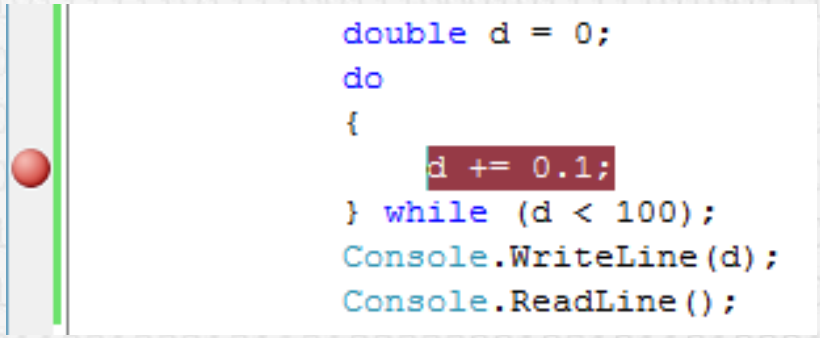
Hibakeresés

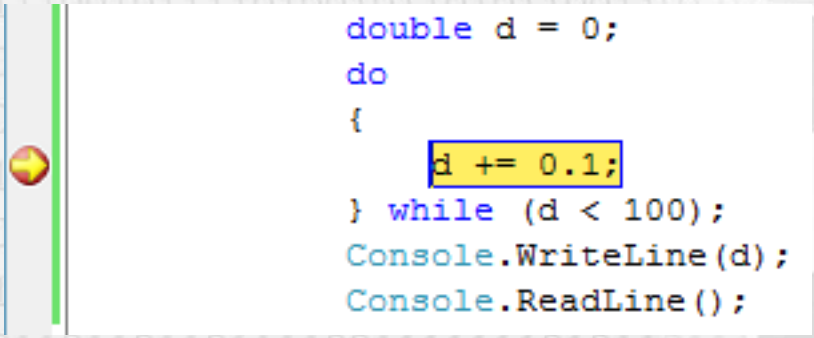
- **Futtatás hibakereséssel (Start Debugging – F5)**



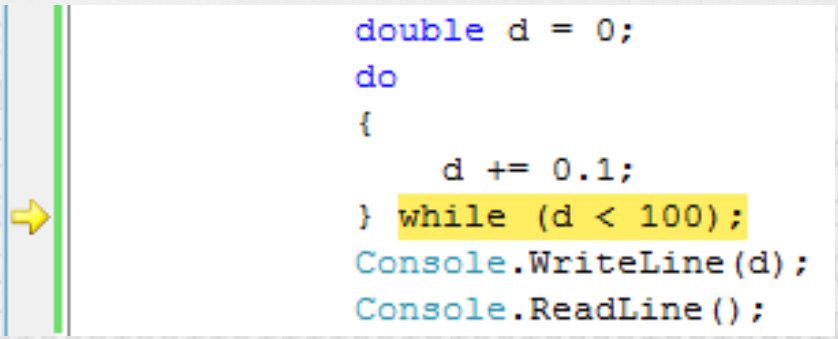
Hibakeresés

- **Töréspont (Breakpoint)**
  - Itt megáll a program végrehajtása.



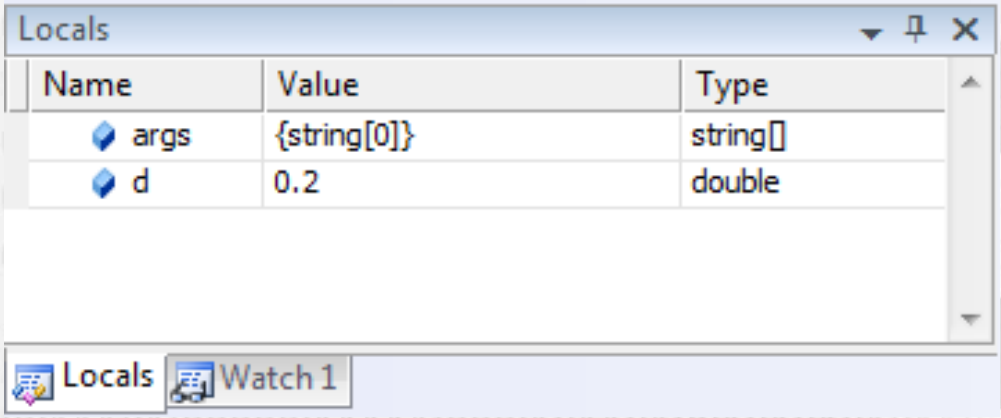


- **Futtatás lépésenként**
  - Step Into (F11)
  - Step Over (F10)

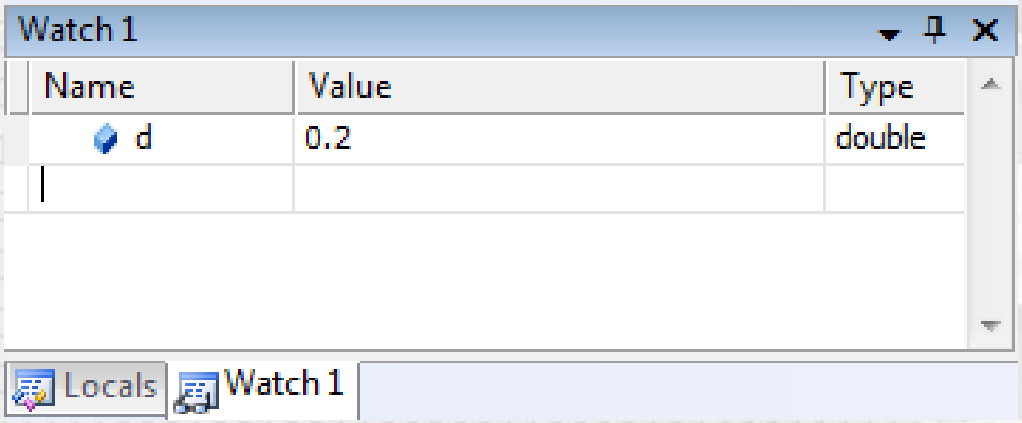


Hibakeresés

- **Locals**
  - Helyi változók: az aktuális blokkban deklarált változók értéke



- **Watch**
  - Megfigyelt változók



Objektumorientált Programozás V.

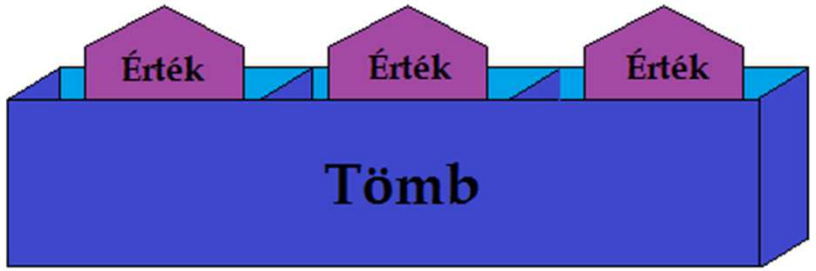
A Microsoft Visual Studio 2010 használata

Műveletek tömbökkel

Érték- és referenciatípusú változók

Feladatok

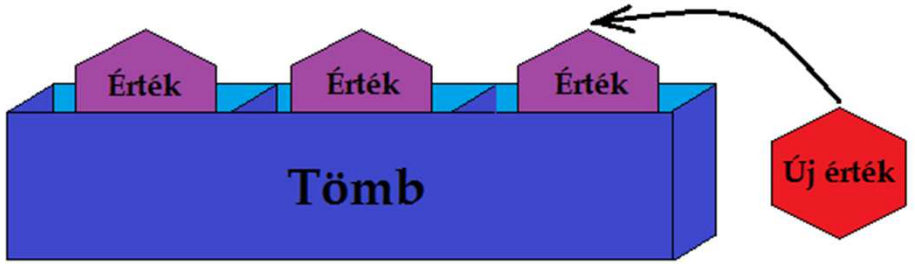
A tömb



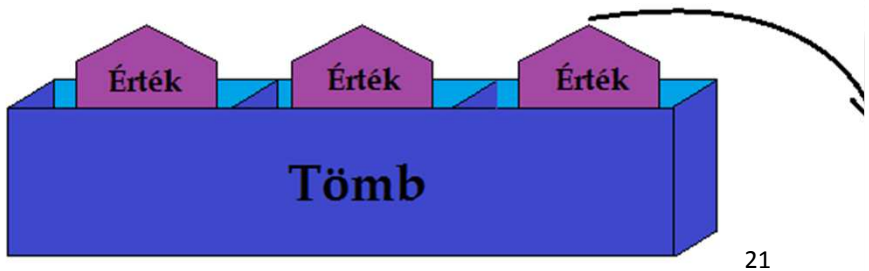
- Több, azonos típusú változó együttes kezelését teszi lehetővé
- int a1; int a2; int a3; int a4; … helyett egyetlen „int tömb” típusú változó használata egy névvel
- Elemei a tömbön belüli sorszámukkal (index) érhetők el (kezdő index: 0!)
- A tömbben tárolt típus és a tömb mérete NEM MÓDOSÍTHATÓ → szigorú adatszerkezet, cserében nagyon gyors

Tömbbel végezhető tevékenységek

- **Deklaráció**
  A tömb nevének és elemei típusának megadása
- **Tömblétrehozás**
  A tömb méretének meghatározása
- **Értékadás**
  Érték elhelyezése egy tömbelemben



- **Érték lekérdezése**
  Egy tömbelem tartalmának kiolvasása.
  Az érték a kiolvasás után is megmarad a tömbelemben



Tömbbel végezhető tevékenységek C#-ban

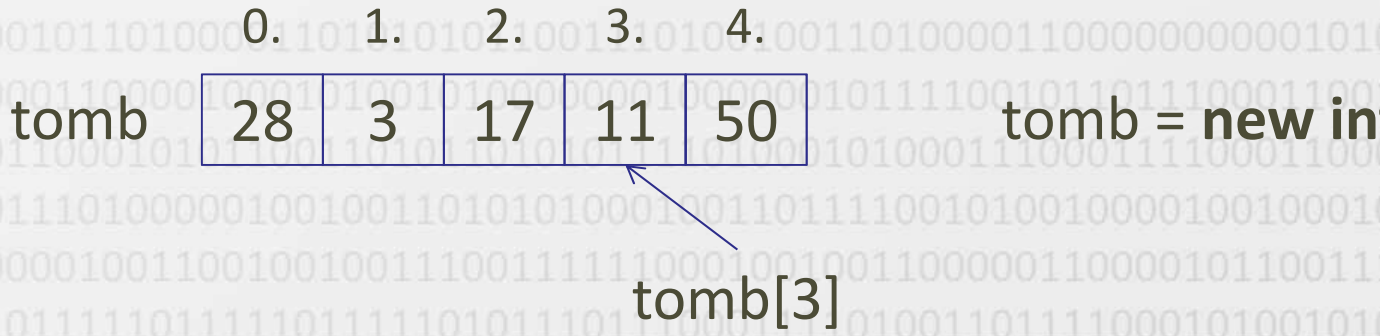
1. Deklaráció
   int[] tomb**;**
2. Tömblétrehozás
   tomb = **new int**[10];
3. Értékadás
   tomb[5] = 25; vagy tomb[5] = 6 * 2 – 29;
4. Érték lekérdezése
   5 * 10 – tomb[5] + 2

A deklaráció és a tömblétrehozás összevonható:

int[] tomb = **new int**[10];

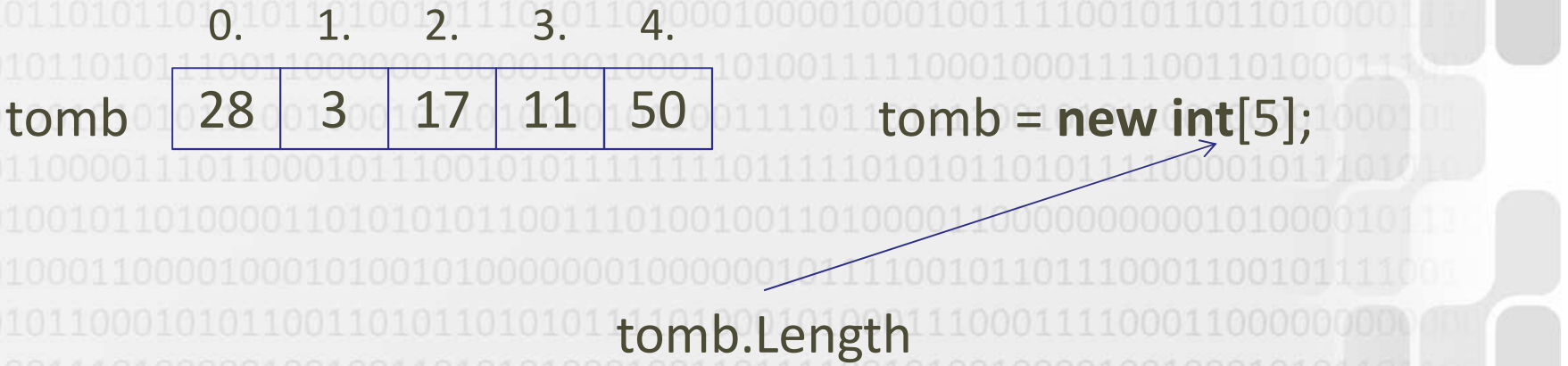
Tömbelem elérése (indexelés)

- **A tömb egy adott eleméhez a tömb neve után szögletes zárójelek között megadott sorszámmal (index) férhetünk hozzá:**
  ***tömbnév*[*index*]**
- **Az index csak egész szám lehet**
- **A tömb első elemének indexe: 0**
- **A tömb utolsó elemének indexe: elemszám – 1**
- **Kisebb, vagy nagyobb index megadása futási hibát okoz.**



Tömb hosszának (elemei számának) lekérdezése

- **Általános formátum:**
  ***tömbnév*.Length**
- **A tömbben lévő elemek számát adja meg**



Tömb inicializálása

- **A tömb deklarációja, létrehozása és elemeinek megadása egy utasításban is elvégezhető**
- **Formátuma:**
  ***típus tömbnév* = {*elem1, elem2, …, elemN*};**
- **Példa:**
  double[] valosak = {2.0, -3, 5, 8.2, -1234.56};
  bool[] logikai = {true, false, false, false, true, true};

Tömbbel végezhető tevékenységek C#-ban

- **A tevékenységek sorrendje:**
  1. Deklaráció
  2. Tömblétrehozás
  3. Értékadás egy tömbelemnek, vagy egy tömbelem értékének lekérdezése
- **A fentiek közül akármelyik tevékenységet szeretnénk is végrehajtani, előbb a sorrendben őt megelőző tevékenységet kell elvégezni!**
- **A tömbelemeknek nem kötelező értéket adni az érték lekérdezése előtt**
- **Értékadás hiányában a tömbelem a típus alapértékét veszi fel**

Tömbbel végezhető tevékenységek C#-ban (melyik kód helyes?)

```
int[] tomb;
Console.WriteLine(tomb[2]);
```

```
int[] tomb = new int[4];
Console.WriteLine(tomb[-1]);
```

```
int[] tomb = new int[4];
Console.WriteLine(tomb[4]);
```

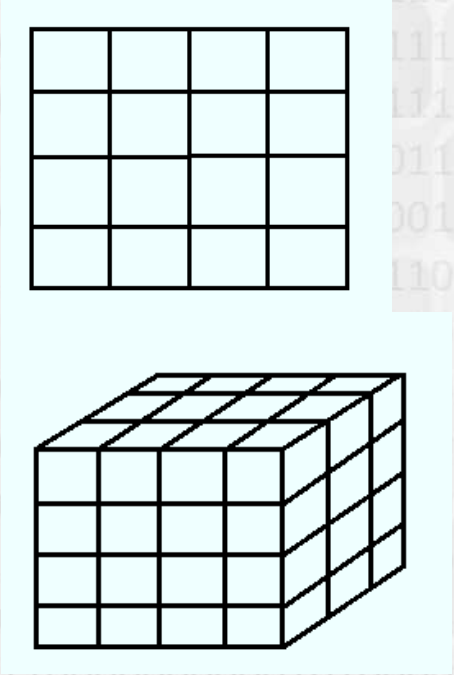
```
int[] tomb = new int[4];
Console.WriteLine(tomb[0]);
```

```
int[] tomb = new int[4];
Console.WriteLine(
        tomb[tomb.Length]);
```

```
int[] tomb = new int[4];
tomb[3] = 22;
Console.WriteLine(tomb[3]);
```

Többdimenziós tömbök

- **2 dimenziós tömb**
  - sorok és oszlopok
  - elem elérése 2 indexszel
- ***3 dimenziós tömb***
  - *sorok, oszlopok, lapok*
  - *elem elérése 3 indexszel*
- ***N dimenziós tömb***
  - *0., 1., … N. dimenzió*
  - *elem elérése N indexszel*



Többdimenziós tömbök – Deklaráció

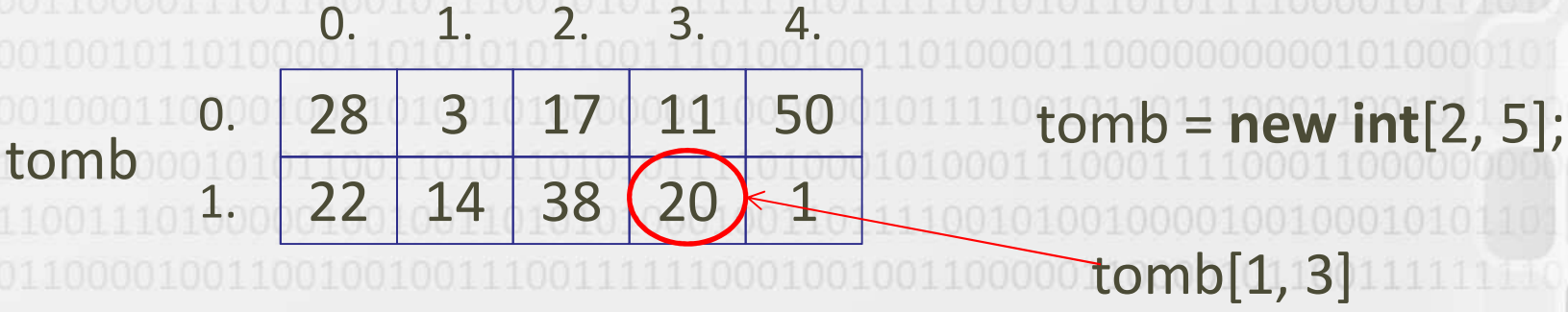
- **Általános formátum:**
  típus*[*vesszők*] *tömbnév;
- **A szögletes zárójelbe dimenziószám-1 darab vesszőt kell tenni**
- **Példák:**
  - **int**[,] matrix;
  - ***bool****[,,] haromdimenziostomb;*
  - ***double****[,,,,] otdimenziostomb;*

Többdimenziós tömbök – Tömblétrehozás

- **Általános formátum:**
  ***tömbnév* = new *tipus* [*elemszám1*, …, *elemszámN*]**
- **Az egyes dimenziók elemszámait vesszőkkel elválasztva kell megadni**
- **A deklaráció és a tömblétrehozás itt is összevonható**
- **Példák**
  - matrix = **new int**[3, 5];
  - haromdimenziostomb = **new bool** [4, 2, 5];
  - **int** t[,,] = **new int**[3, 3, 3];
  - int[,] egeszmatrix = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 0, 1, 2}};

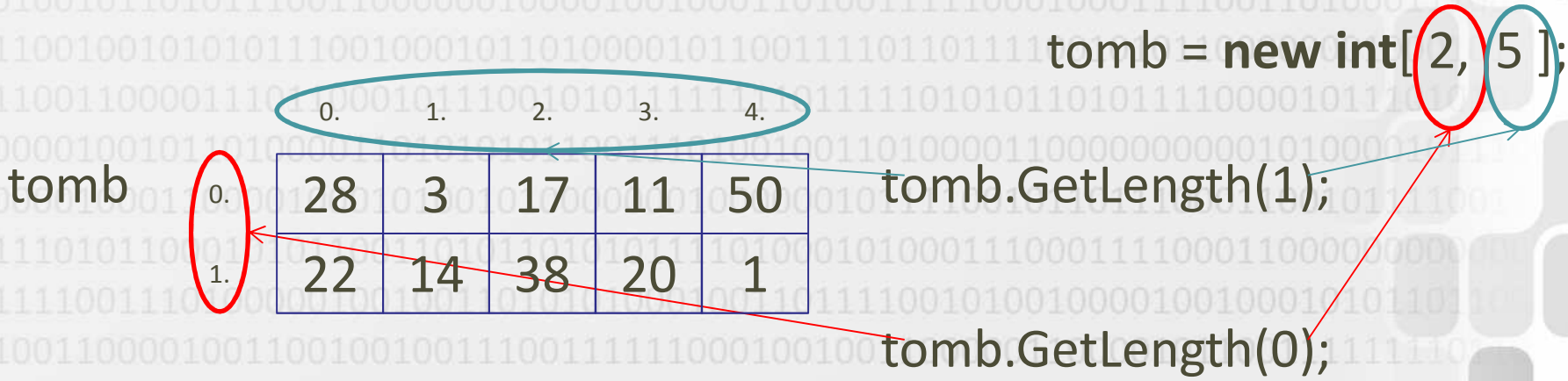
Tömbelem elérése (indexelés)

- **A szögletes zárójelek közé a tömbelem minden egyes dimenzióján belüli sorszámait kell vesszőkkel elválasztva megadni:**
  ***tömbnév*[*index1, index2, …, indexN*]**
- **Az indexekre vonatkozó szabályok u.a., mint az egydimenziós tömbnél**
- **pontosan annyi indexet kell megadni, ahány dimenziós a tömb**



Tömb hosszának (elemei számának) lekérdezése

- **Elemek számának lekérdezése:**
  - Összes tömbben lévő elem darabszáma: *tömbnév*.Length
  - Egy adott dimenzió elemszáma (sorok száma, oszlopok száma, …): *tömbnév*.GetLength(*dimenziósorszám*);



A for utasítás

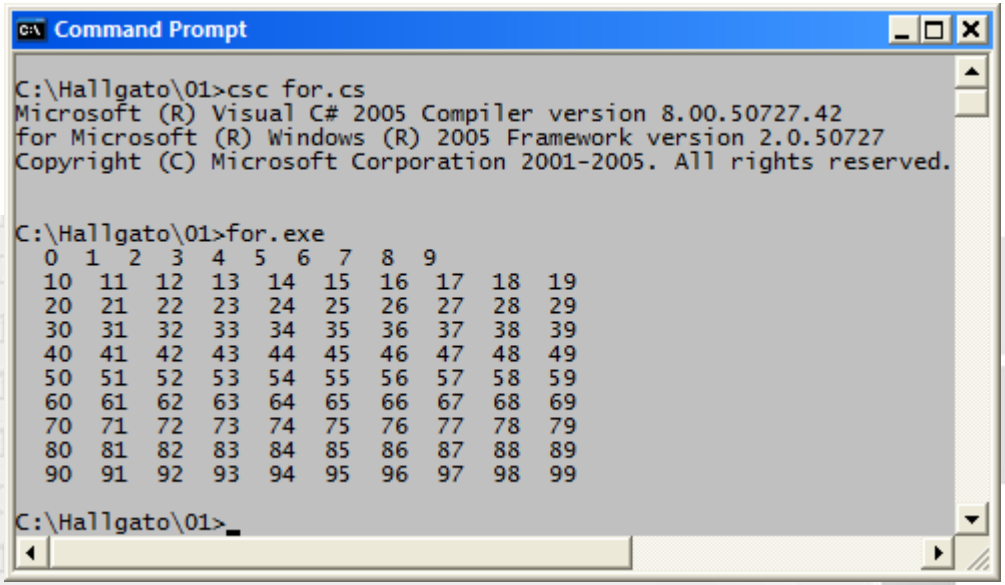
```
for (inicializátor ; feltétel ; iterátor )
   utasítás
```

- **Az inicializátor és az iterátor tetszőleges utasítás lehet**
- **Működése:**
  - Belépéskor egyszer végrehajtódik az inicializátor
  - Minden ciklusmenetben kiértékelődik a feltétel
  - Amennyiben a feltétel igaz, az utasítás ciklusmag egyszer lefut
  - A ciklusmag végeztével végrehajtódik az iterátor és ismét kiértékelődik a feltétel
  - A ciklus akkor ér véget, amikor a feltétel hamissá válik, ellenkező esetben újabb ciklusmenet következik
- **Általában az inicializátor egy számlálót állít be, az iterátor pedig ezt a számlálót növeli vagy csökkenti**
  - Legtöbbször akkor használjuk, ha előre ismert számú alkalommal szeretnénk végrehajtani egy utasítást

A for utasítás (példa)

```
// Számmátrix

// Ez a külső ciklus fut végig az összes soron
for (int i = 0; i < 100; i += 10)
{
    // Ez a belső ciklus fut végig egy soron belül az összes oszlopon
    for (int j = i; j < i + 10; j++)
    {
        System.Console.Write("  " + j);
    }
    System.Console.WriteLine();
}
```



A foreach

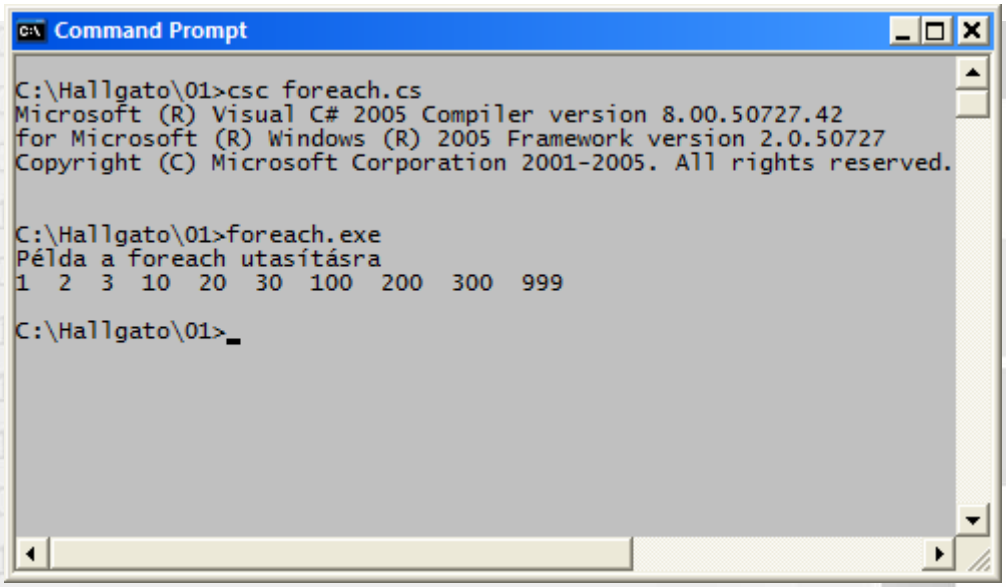
foreach (típus változó in gyűjtemény)
utasítás

- **Lehetővé teszi egy utasítás végrehajtását egy adott gyűjtemény összes elemére**
  - A „gyűjtemény” pontos fogalmát később részletesen tárgyaljuk
  - A tömbök gyűjtemények, tehát a foreach utasítás használható hozzájuk
- **Működése:**
  - Belépéskor létrejön egy „típus” típusú változó („iterációs változó”)
    - Ez a változó csak az utasításon belül használható
  - Az utasítás annyiszor hajtódik végre, ahány elemet tartalmaz a gyűjtemény
  - Az iterációs változó minden egyes végrehajtásnál felveszi a gyűjtemény soron következő elemének értékét
- **Az iterációs változó az utasításban nem módosítható**
  - Erre a célra a for utasítás használható

A foreach (példa)

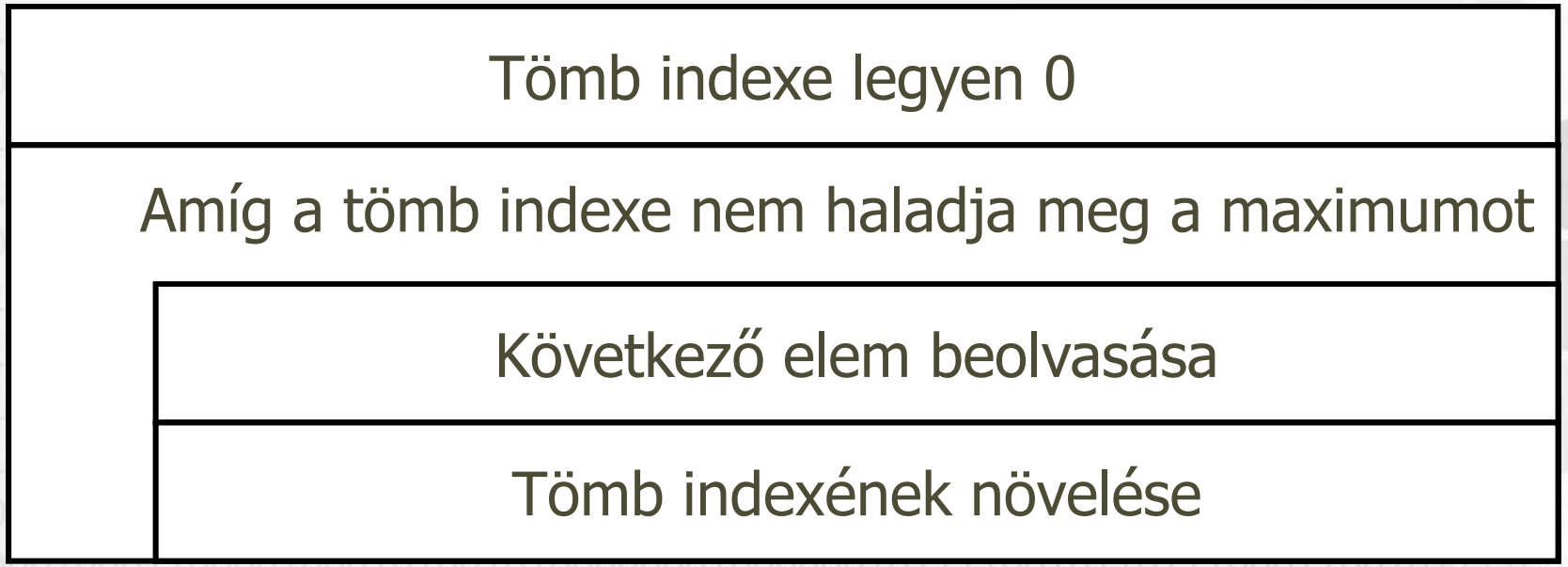
```
int[] teszttömb = {1, 2, 3, 10, 20, 30, 100, 200, 300, 999};

System.Console.WriteLine("Példa a foreach utasításra");
foreach (int tömbérték in teszttömb)
{
   System.Console.Write(tömbérték + "  ");
}
System.Console.WriteLine();
```



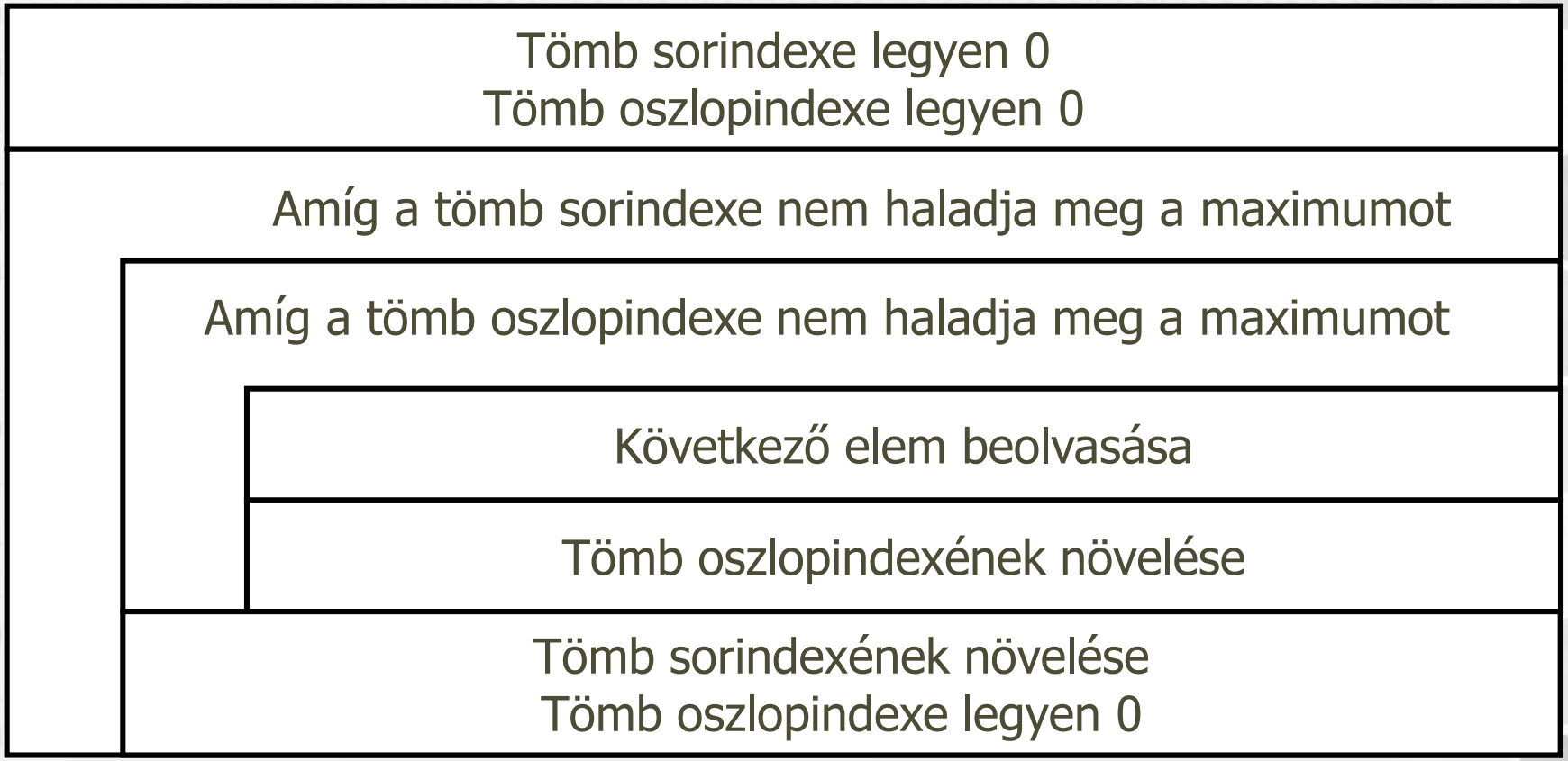
Gyakorló feladatok

Készítsünk algoritmust, majd programot, amely elvégzi egy egydimenziós tömb feltöltését a konzolról beolvasott adatokkal!



Gyakorló feladatok

Készítsünk algoritmust, majd programot, amely a konzolról beolvassa egy kétdimenziós, 3x3-as tömb minden elemét, majd kiírja a tömb teljes tartalmát!



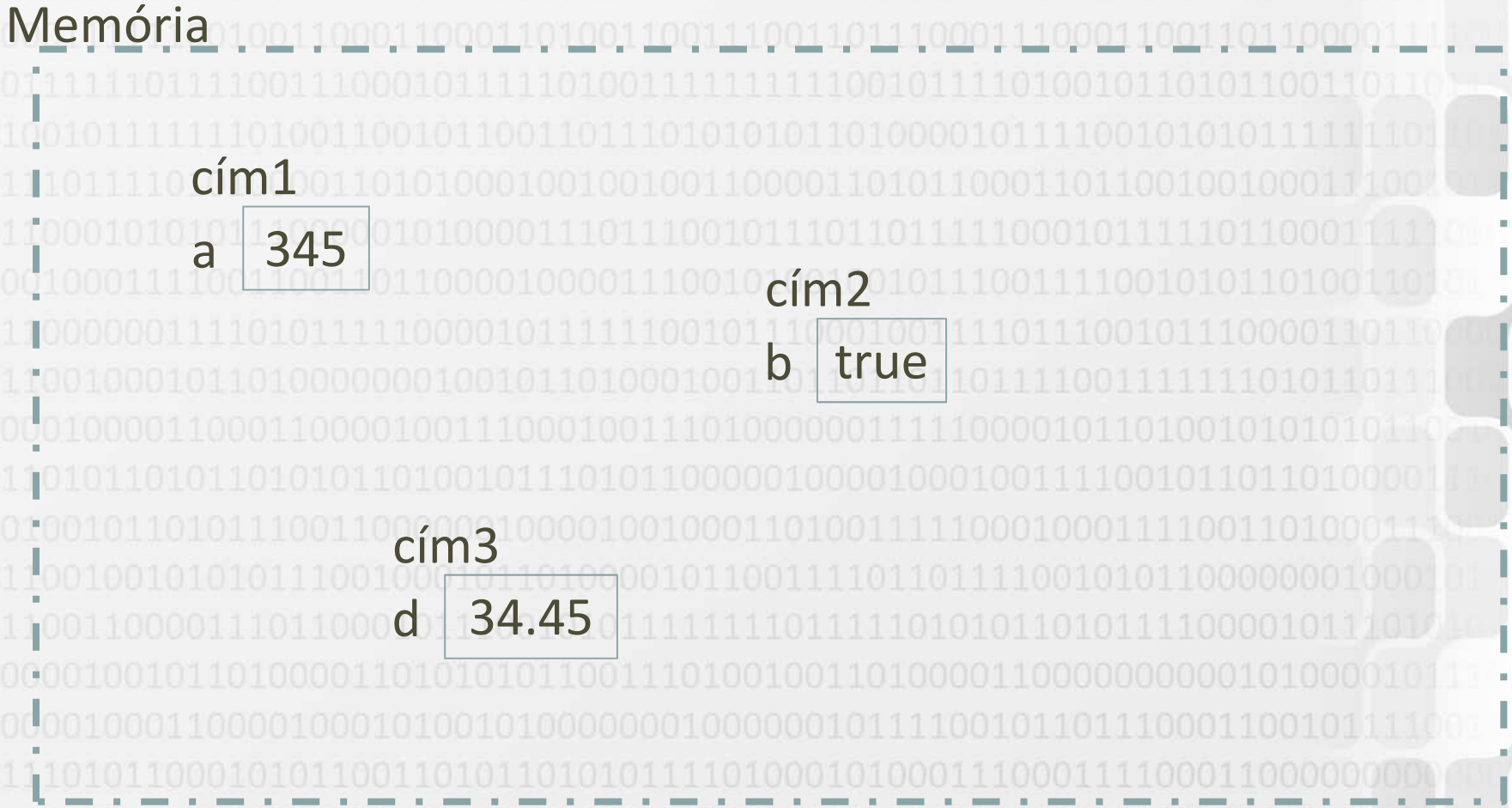
Objektumorientált Programozás V.

A Microsoft Visual Studio 2010 használata
Műveletek tömbökkel
Érték- és referenciatípusú változók
Feladatok

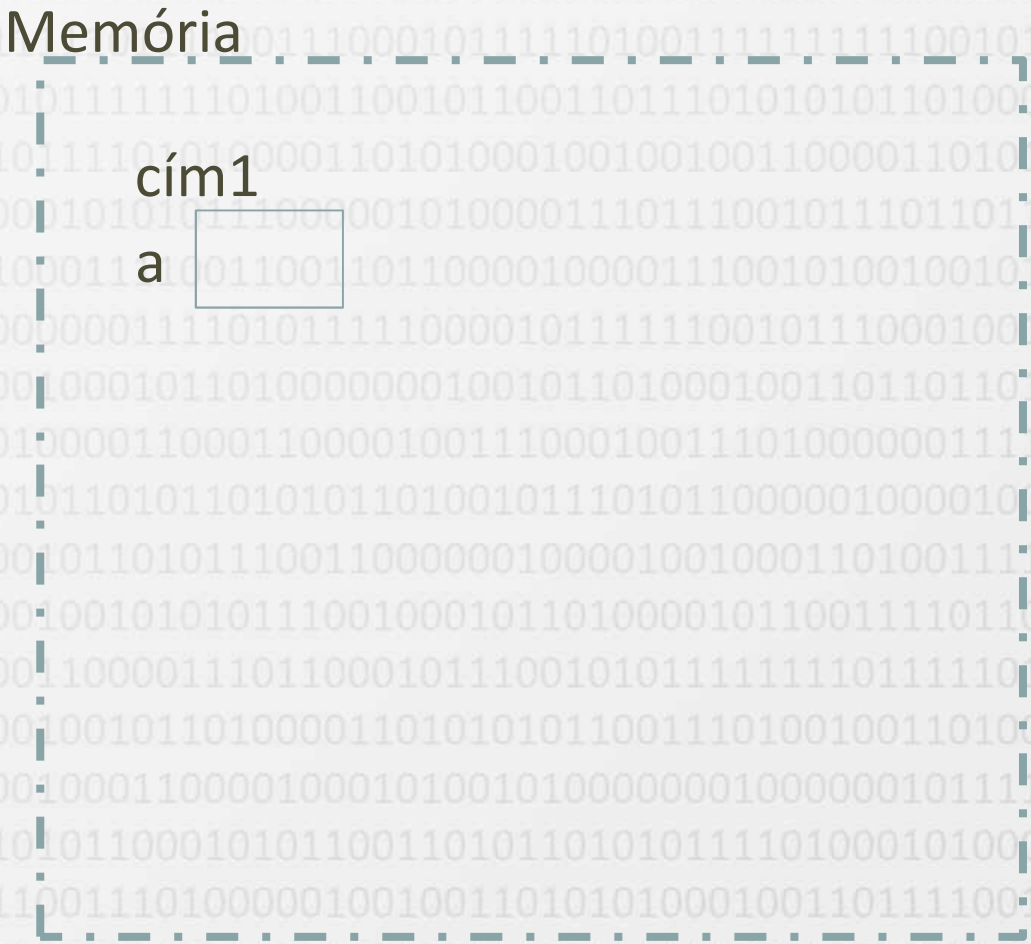
Érték- és referenciatípusú változók

- **Értéktípusú változók**
  - Konkrét értéket tárolnak
  - Az "a = b" értékadáskor a b értéke átmásolódik a-ba
  - egész, valós, logikai, karakter, felsorolás, struktúra típusok
- **Referenciatípusú változók**
  - Csak egy hivatkozást tárolnak a konkrét értékre
  - Az "a = b" értékadáskor a cím másolódik → b ugyanazon memóriacímen lévő értékre fog hivatkozni, mint a.
  - tömb, osztály, interfész, delegált, esemény típusok

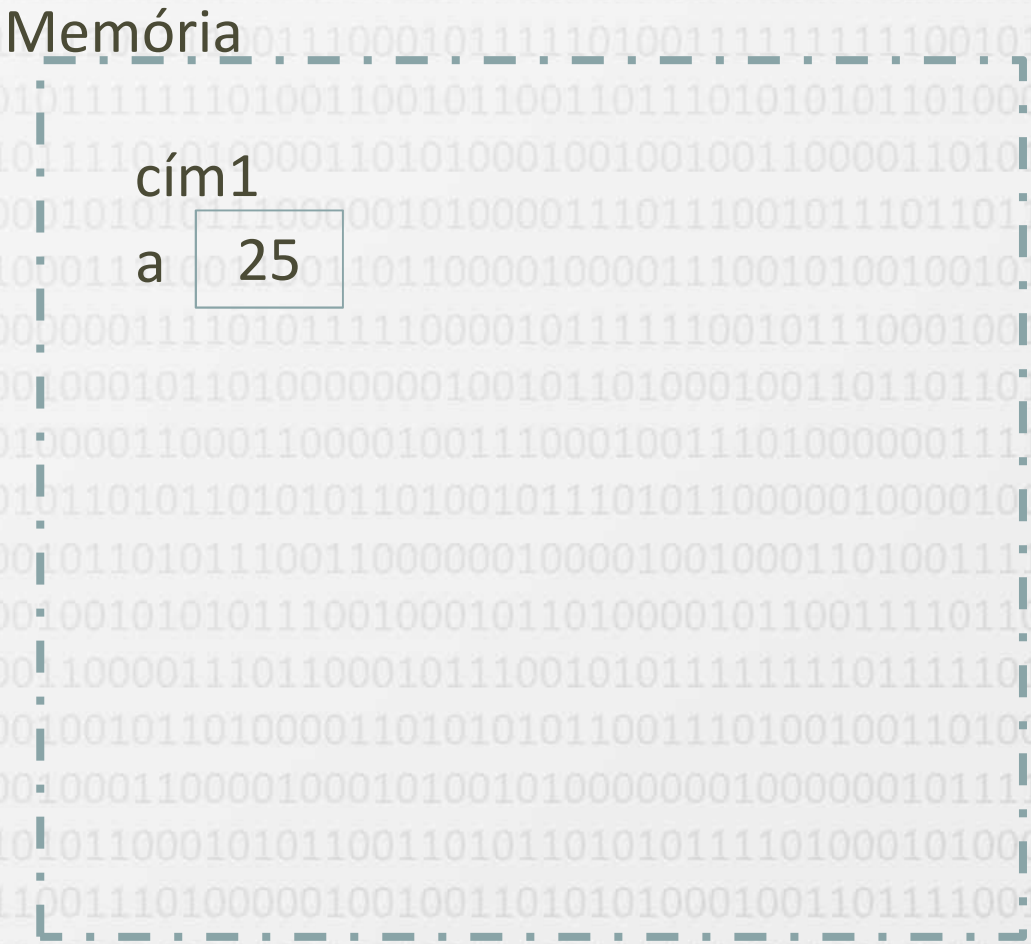
Változók a memóriában (sematikus ábra)



Érték típusú változók a memóriában 1. (sematikus ábra)

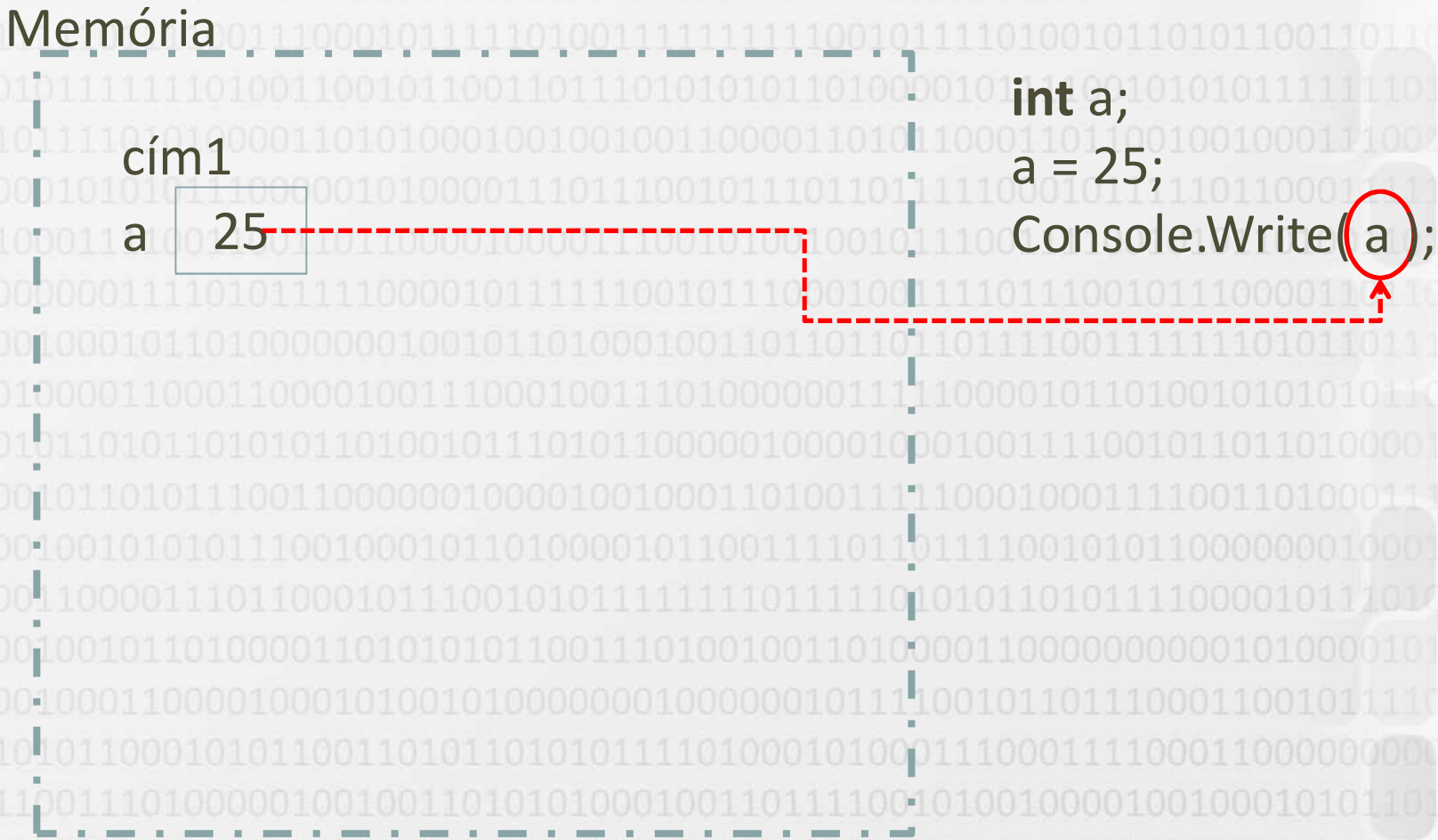


Érték típusú változók a memóriában 2. (sematikus ábra)

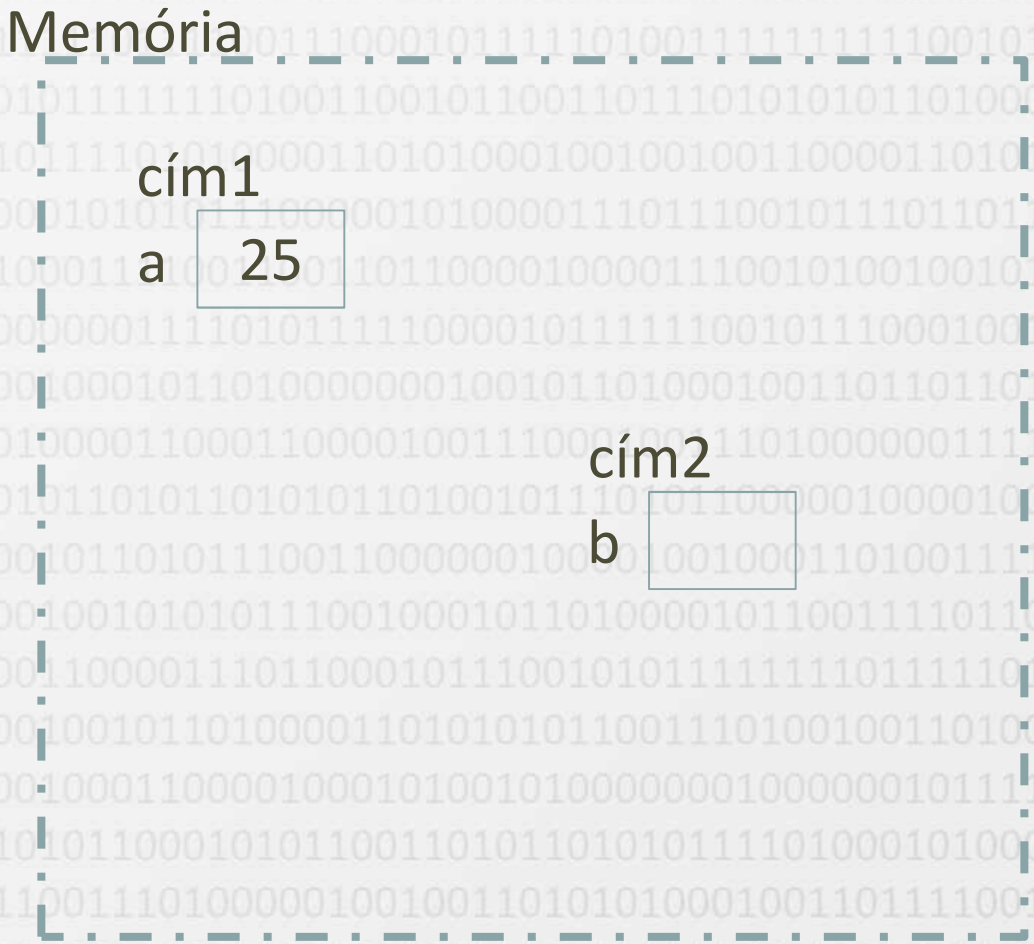



```
int a;
a = 25;
```

Érték típusú változók a memóriában 3. (sematikus ábra)

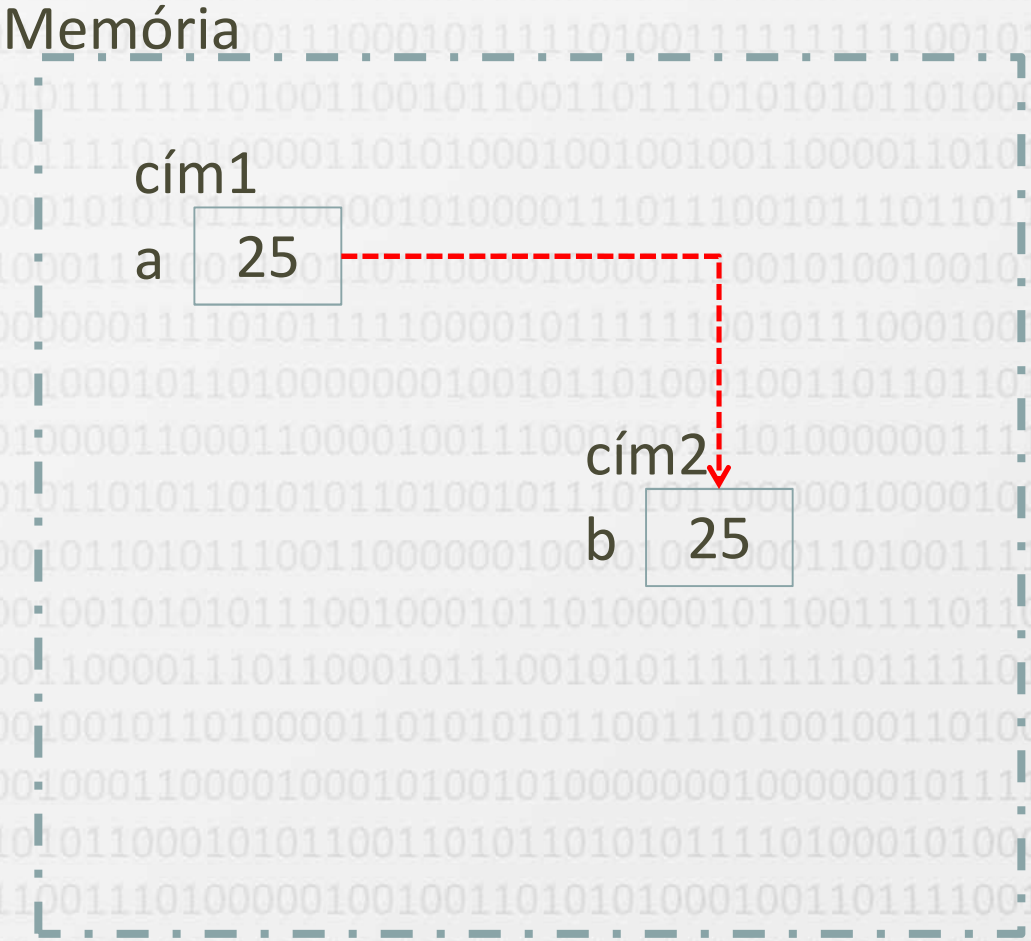


Érték típusú változók a memóriában 4. (sematikus ábra)



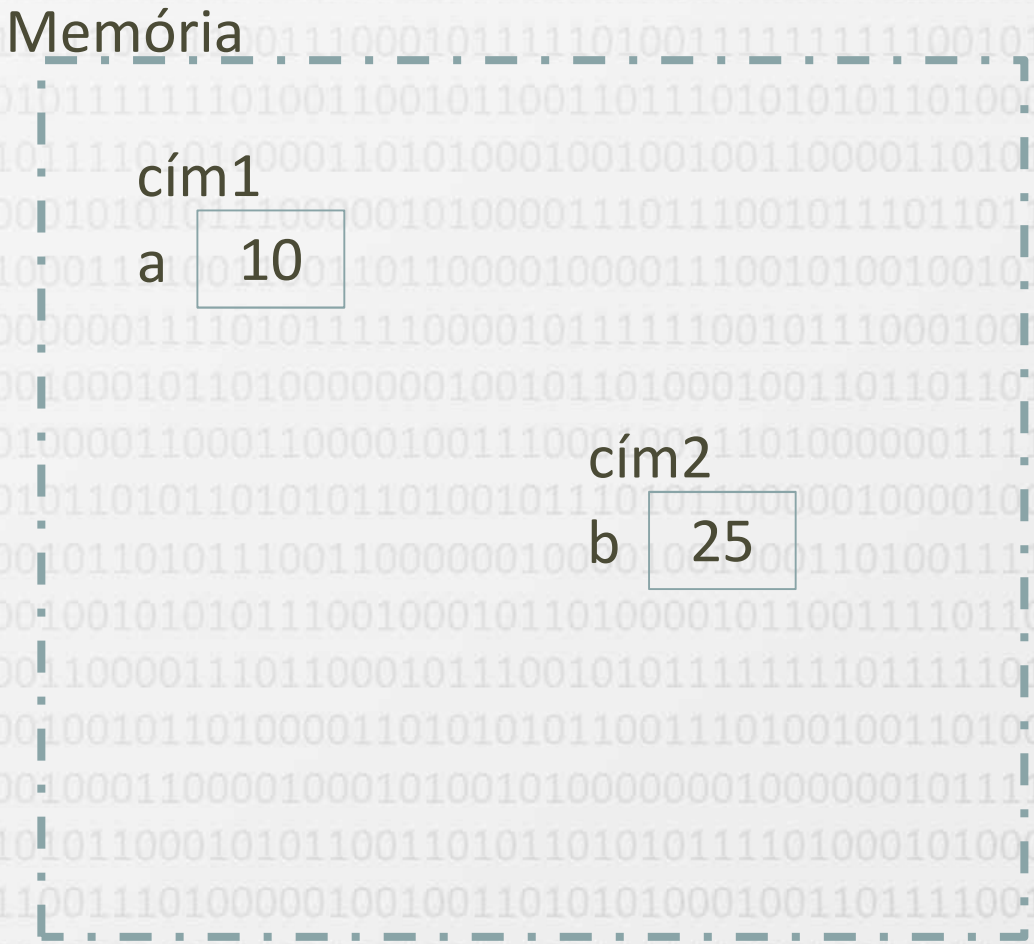

```
int a;
a = 25;
Console.Write( a );
int b;
```

Érték típusú változók a memóriában 5. (sematikus ábra)



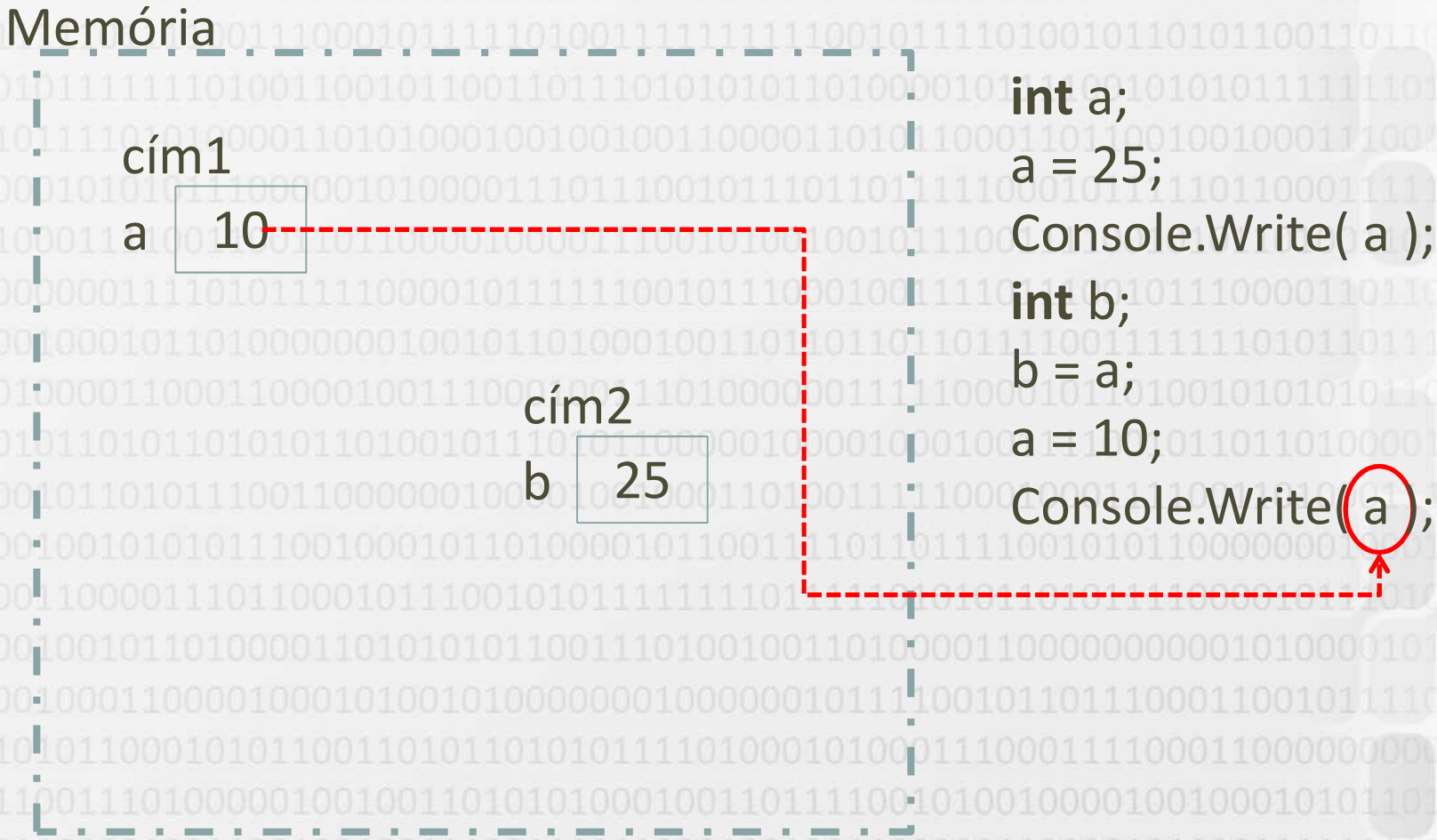

```
int a;
a = 25;
Console.Write( a );
int b;
b = a;
```

Érték típusú változók a memóriában 6. (sematikus ábra)

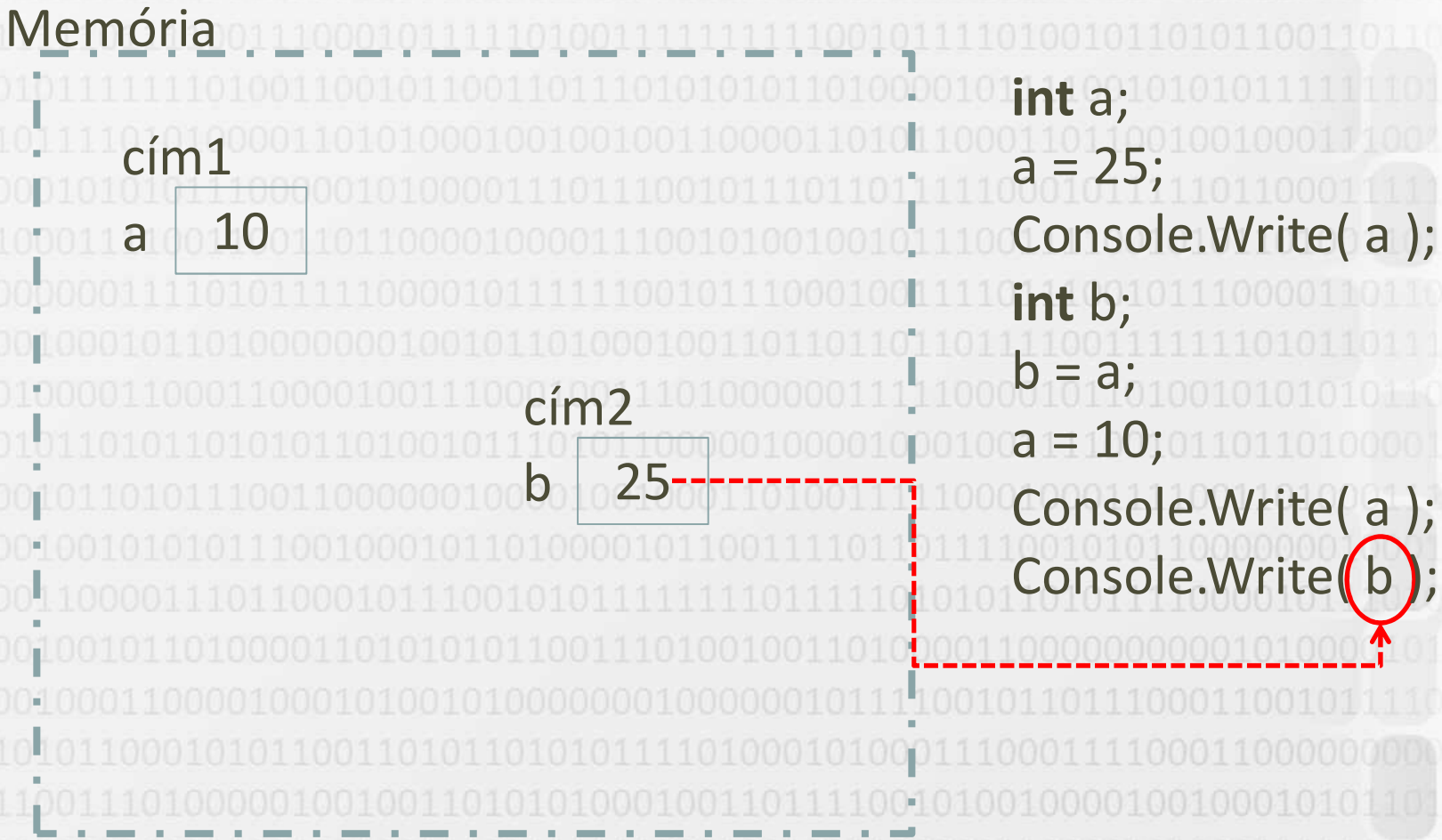



```
int a;
a = 25;
Console.Write( a );
int b;
b = a;
a = 10;
```

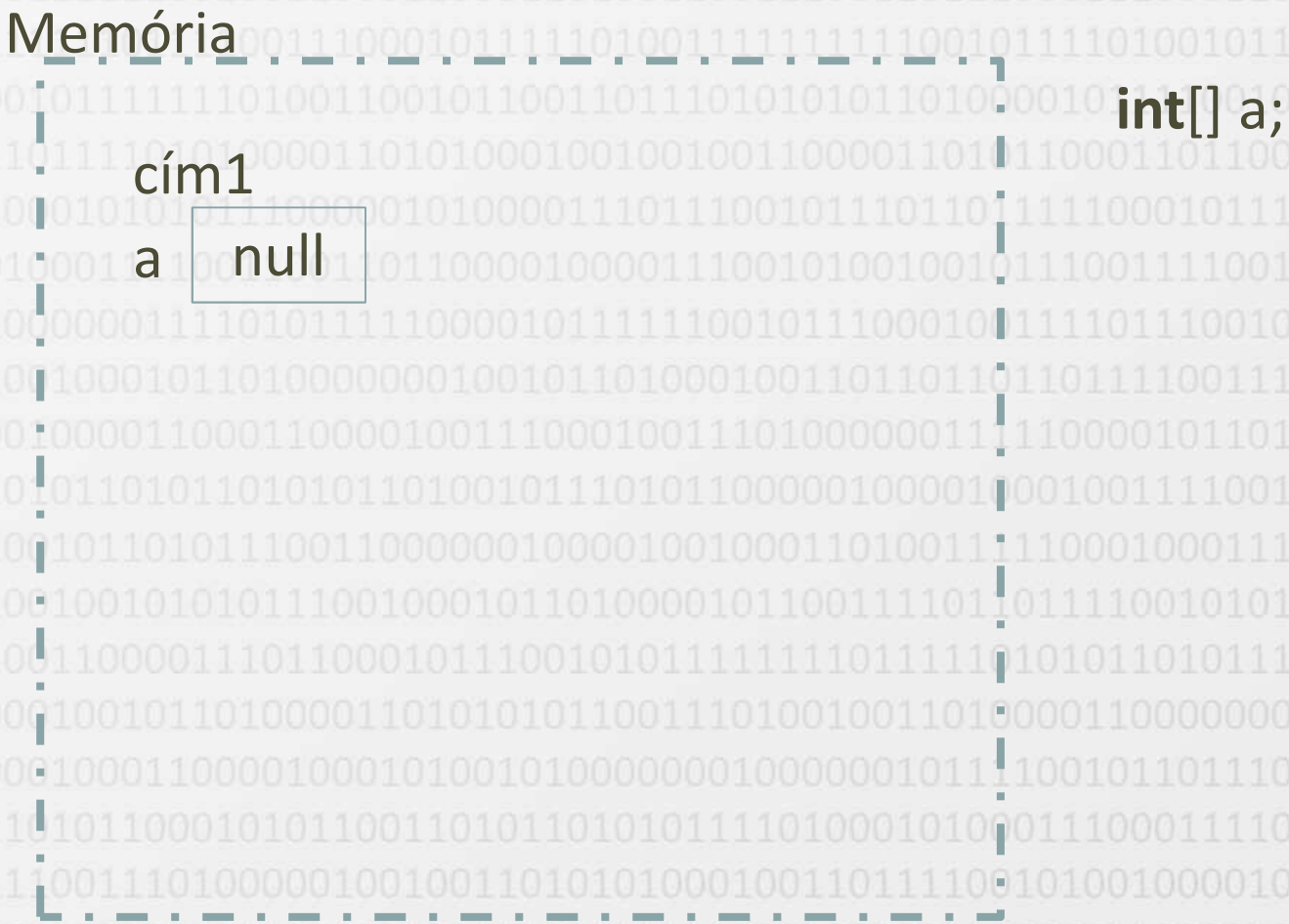
Érték típusú változók a memóriában 7. (sematikus ábra)



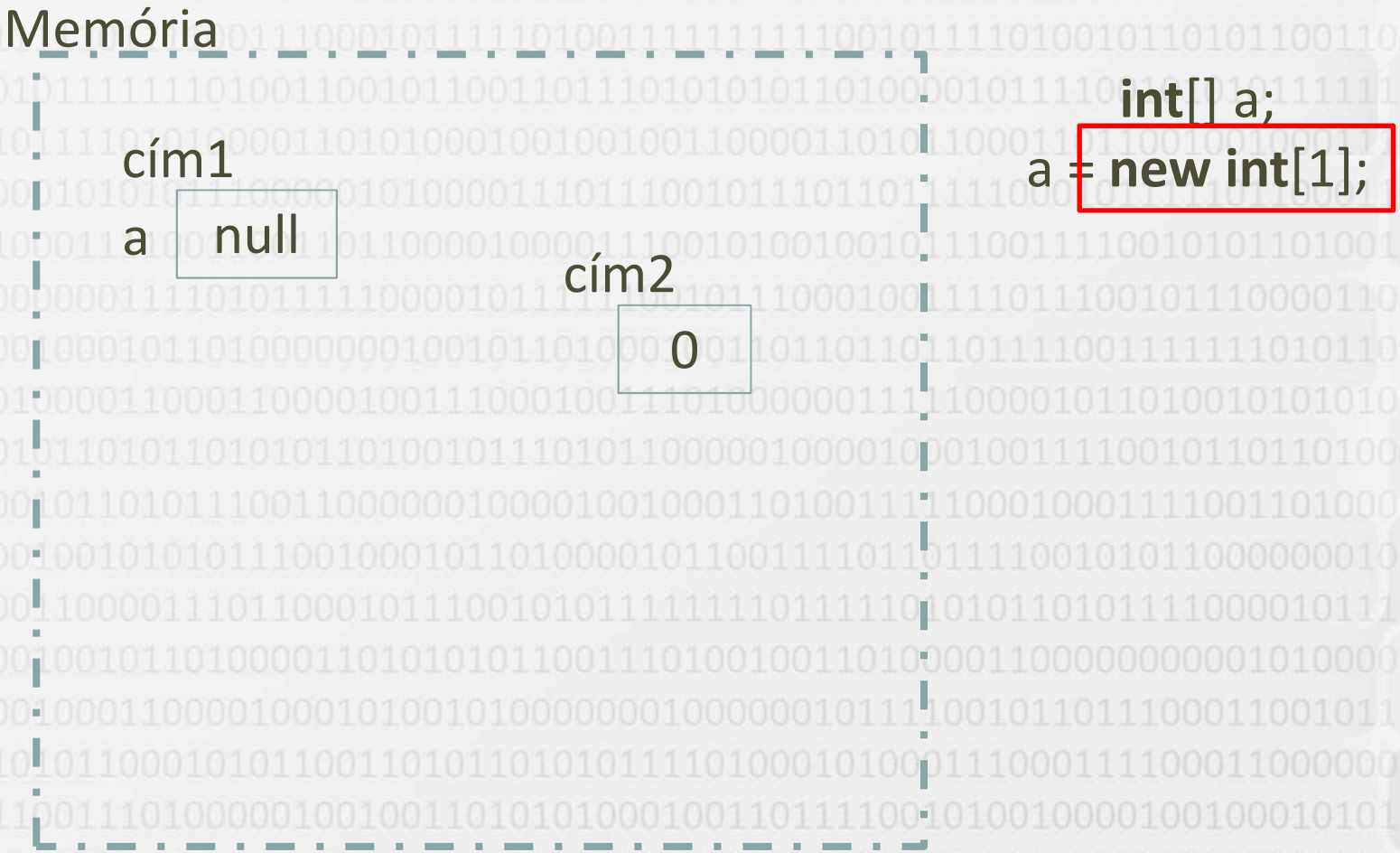
Érték típusú változók a memóriában 8. (sematikus ábra)



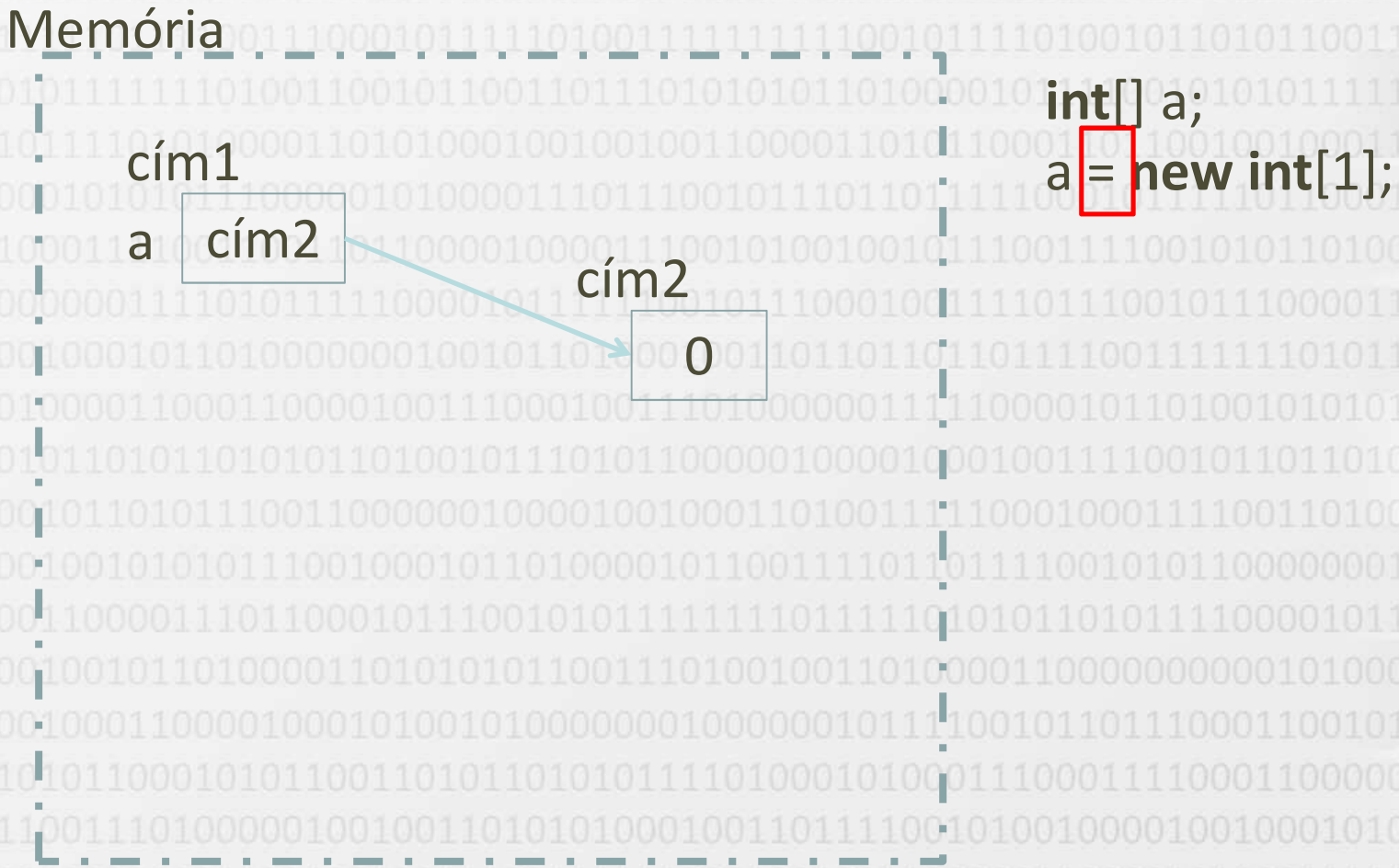
Referencia típusú változók a memóriában 1. (sematikus ábra)



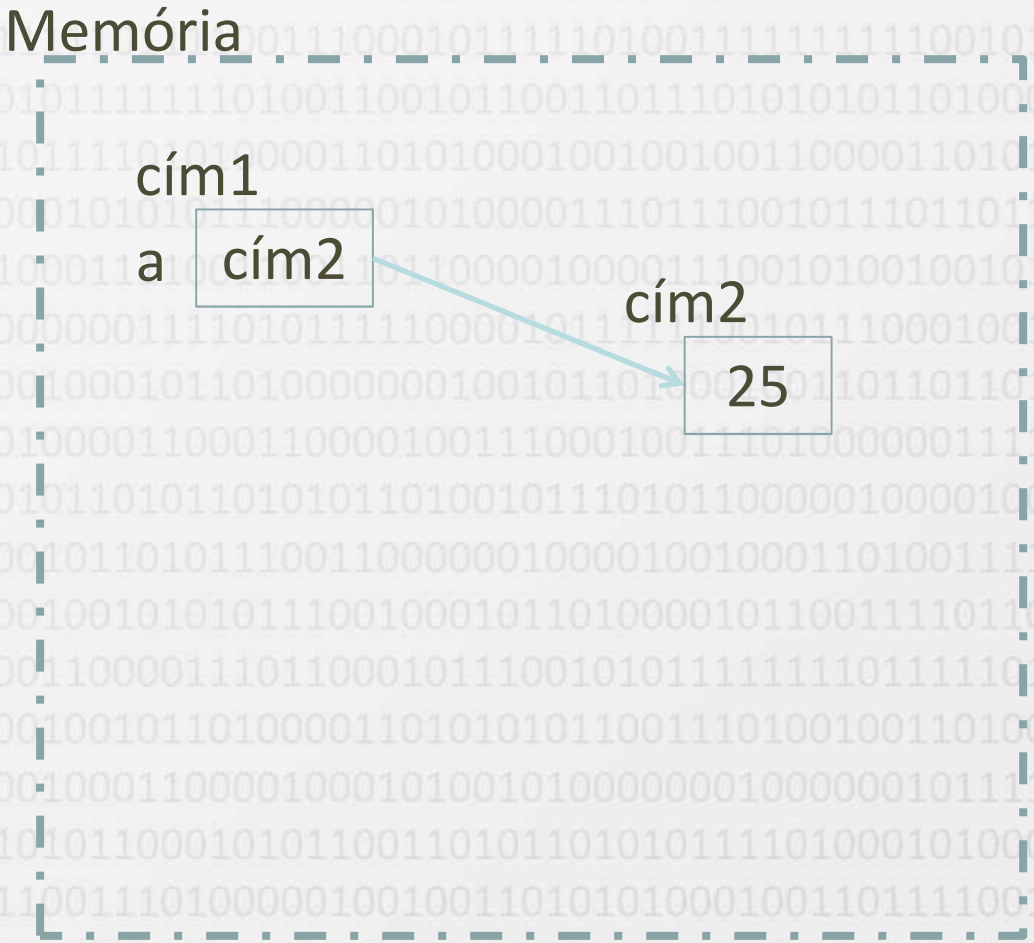
Referencia típusú változók a memóriában 2. (sematikus ábra)



Referencia típusú változók a memóriában 3. (sematikus ábra)

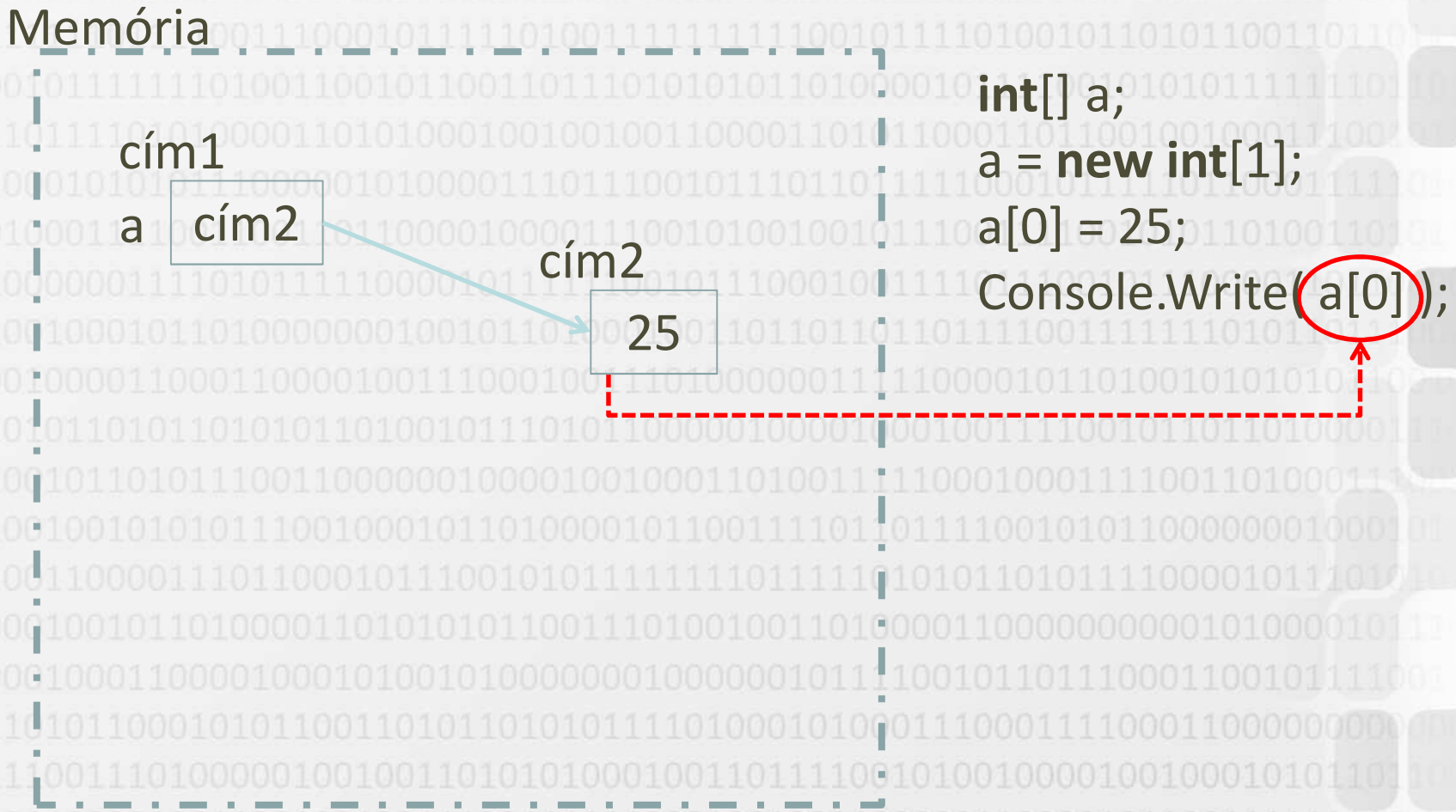


Referencia típusú változók a memóriában 4. (sematikus ábra)

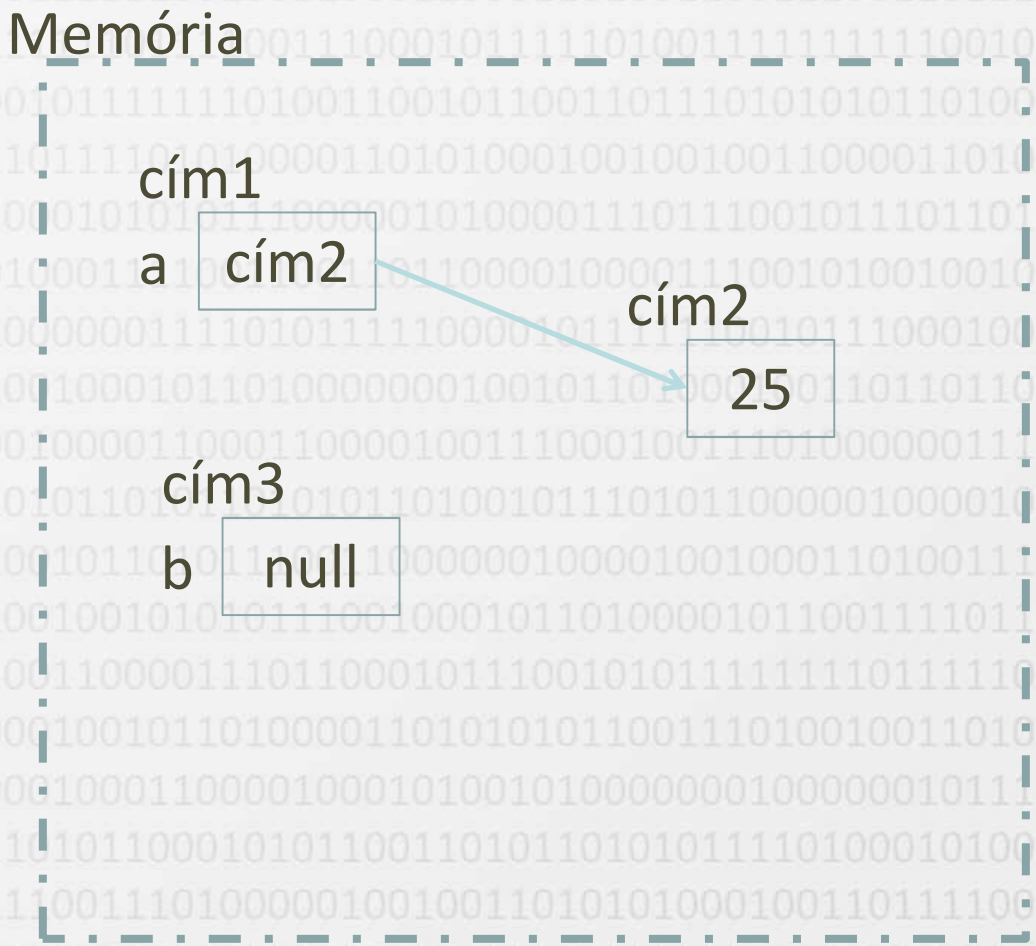



```
int[] a;
a = new int[1];
a[0] = 25;
```

Referencia típusú változók a memóriában 5. (sematikus ábra)

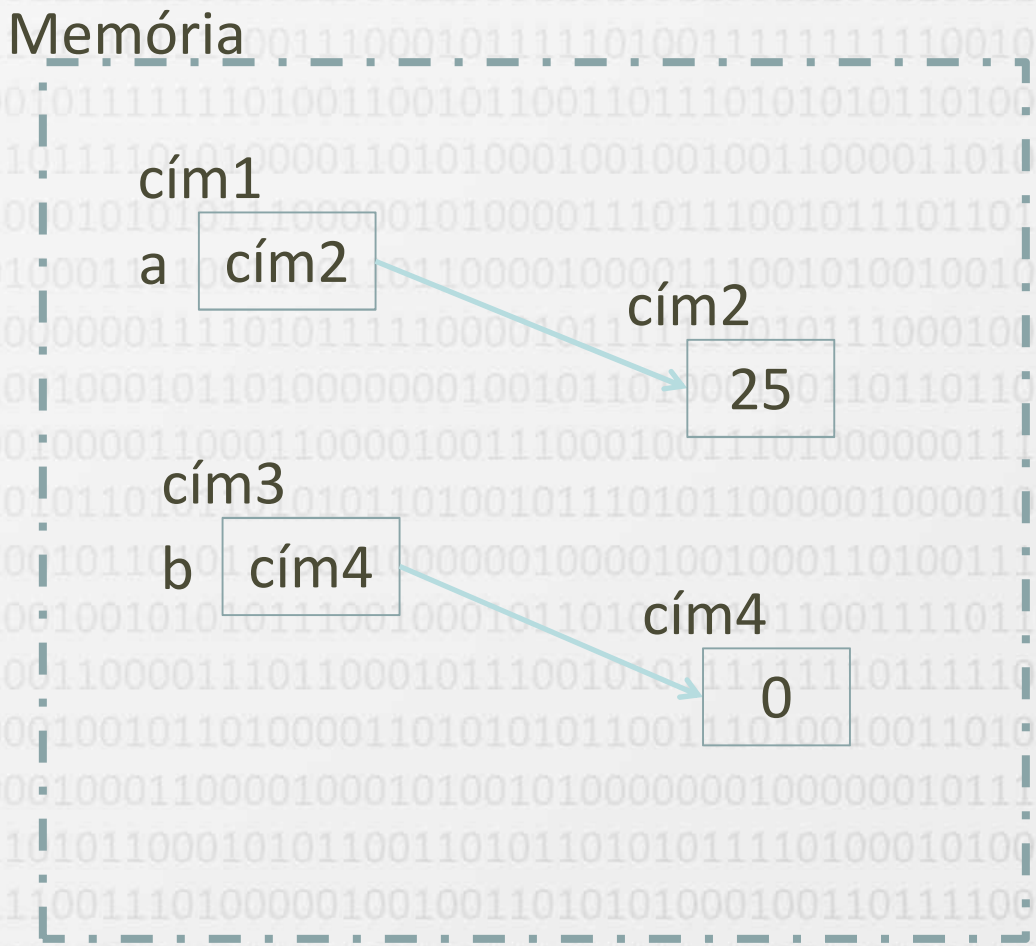


Referencia típusú változók a memóriában 6. (sematikus ábra)



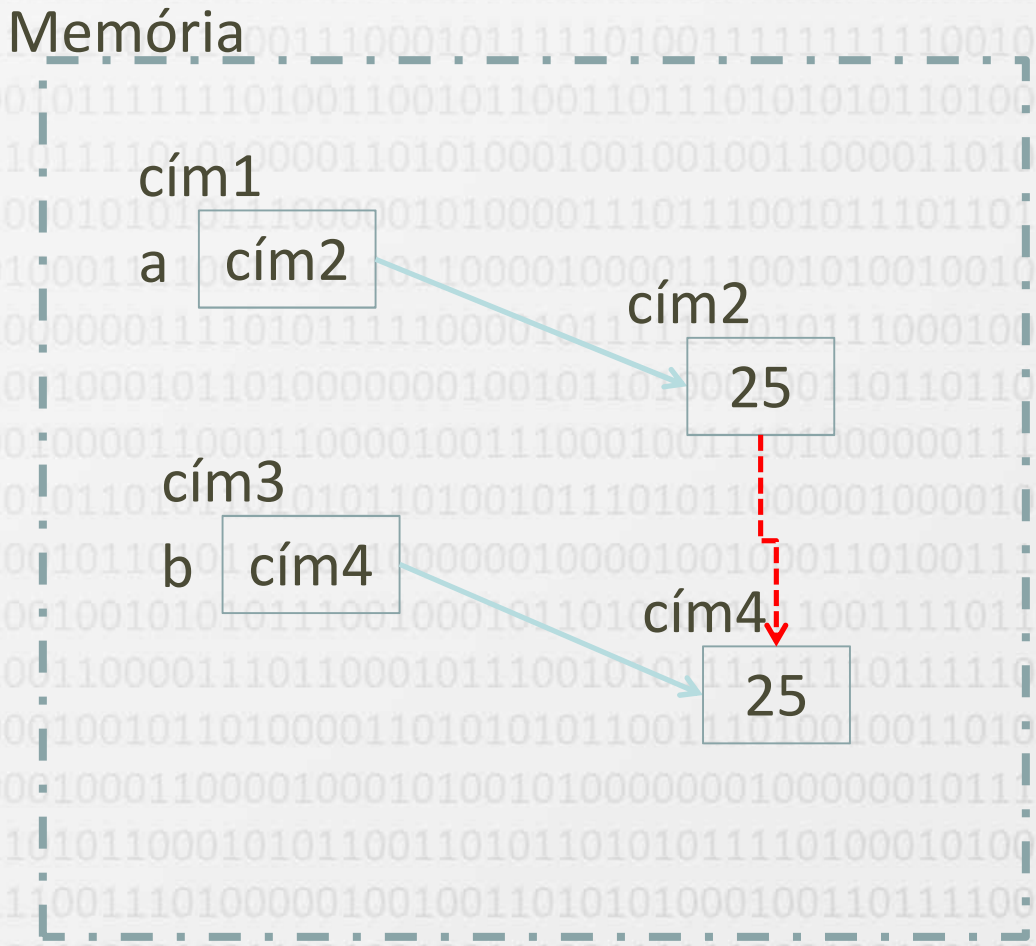

```
int[] a;
a = new int[1];
a[0] = 25;
Console.Write( a[0] );
int[] b;
```

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];
```

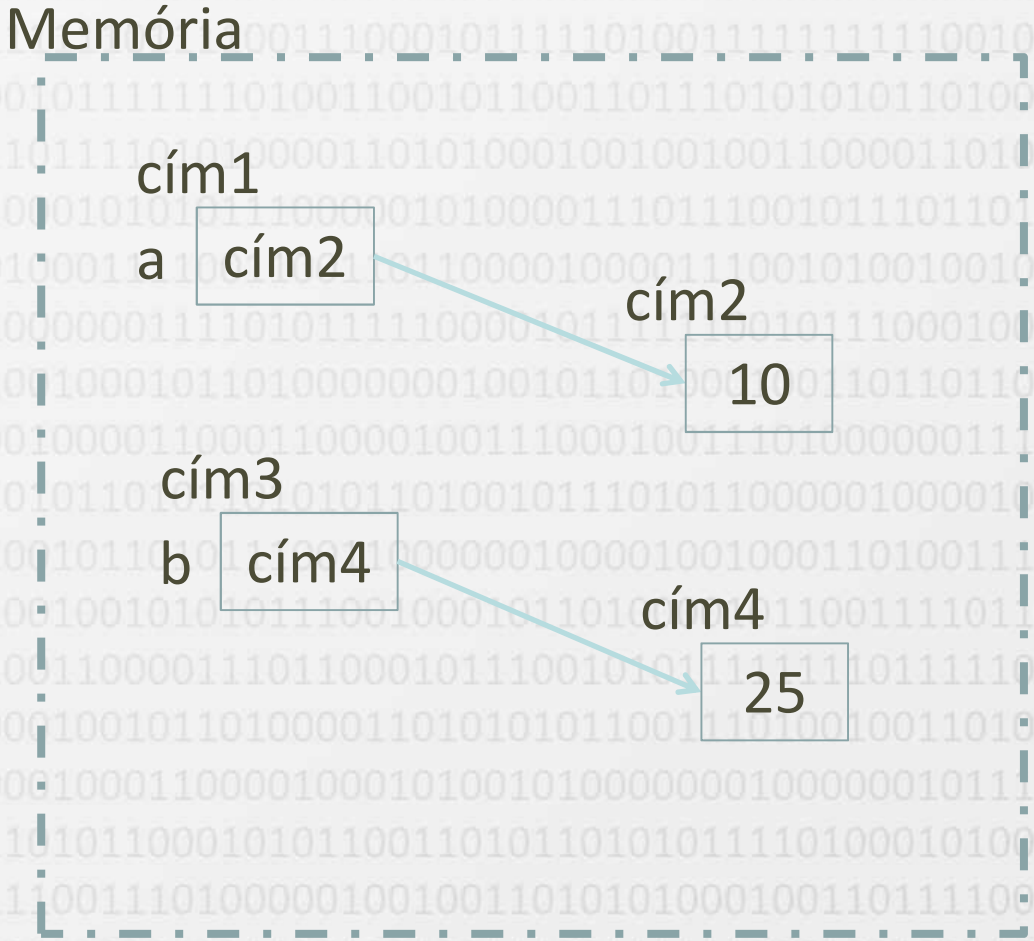
Referencia típusú változók a memóriában 8. (sematikus ábra)



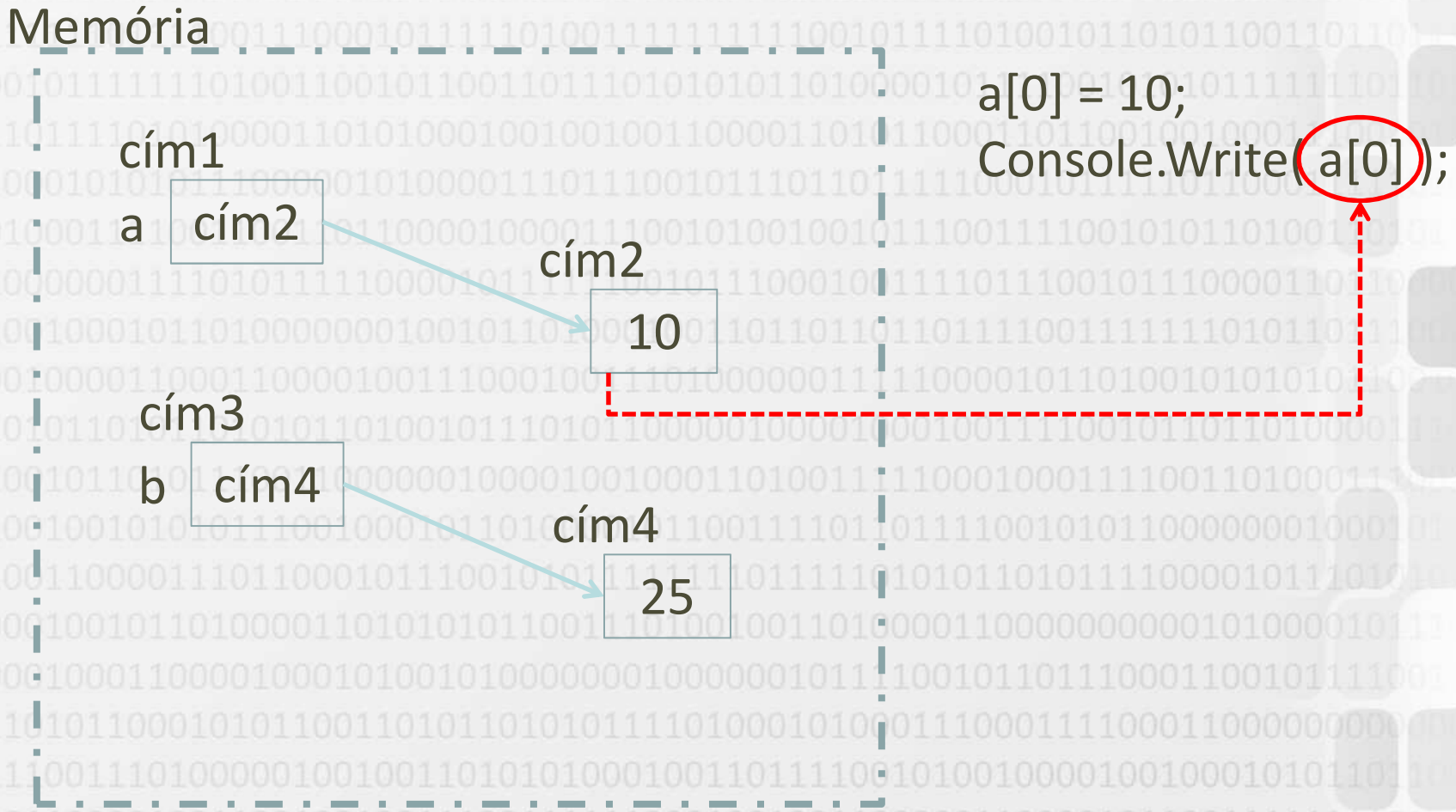

```
int[] a;
a = new int[1];
a[0] = 25;
Console.Write( a[0] );
int[] b;
b = new int[1];
b[0] = a[0];
```

Ez még érték típusú változók értékadása!

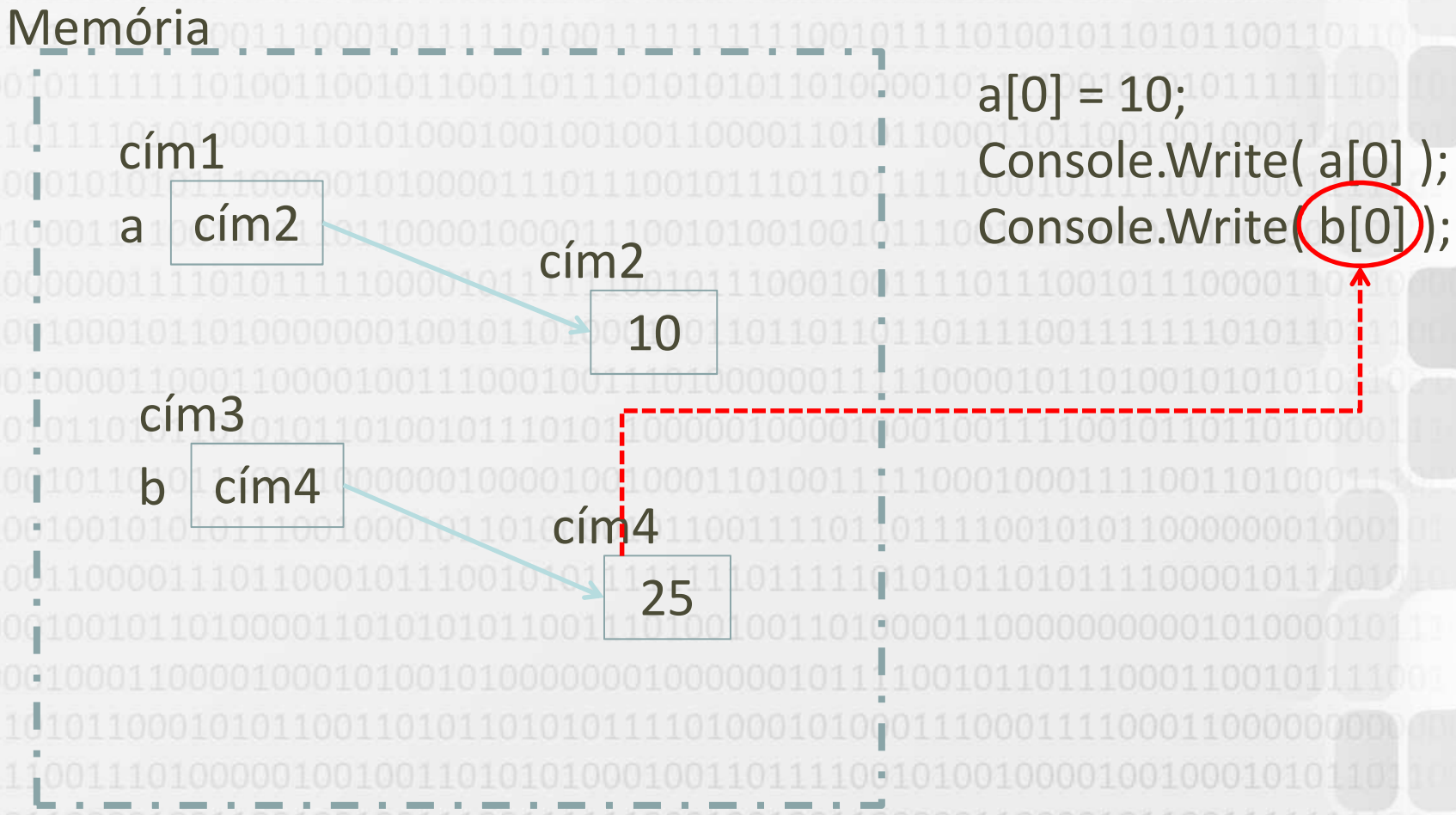
Referencia típusú változók a memóriában 9. (sematikus ábra)



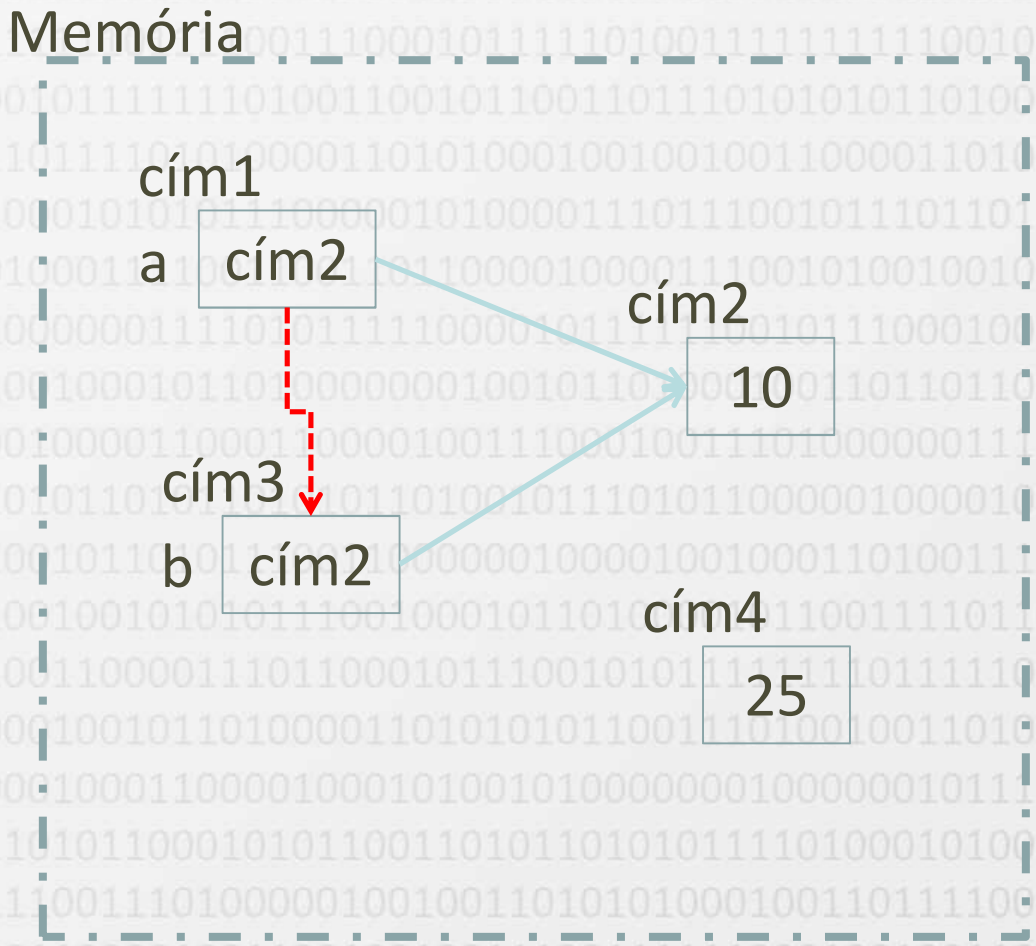
Referencia típusú változók a memóriában 10. (sematikus ábra)



Referencia típusú változók a memóriában 11. (sematikus ábra)



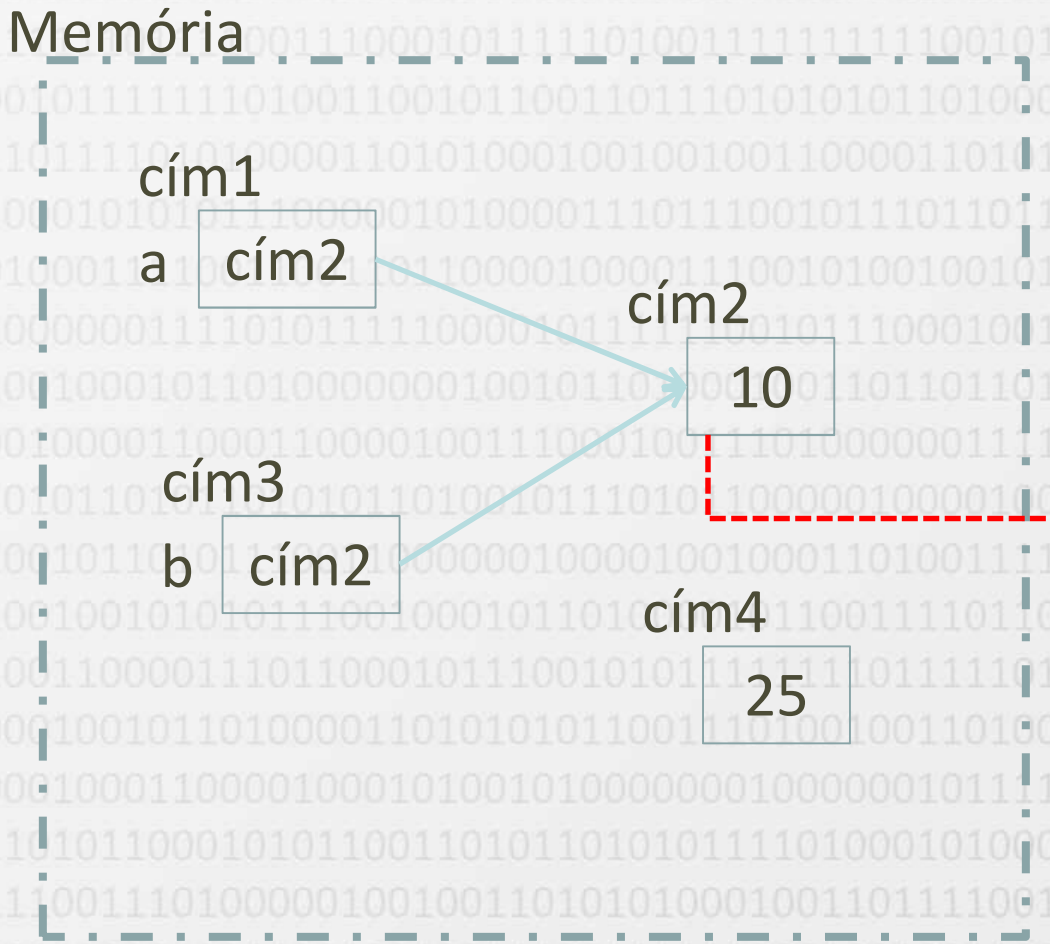
Referencia típusú változók a memóriában 12. (sematikus ábra)




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

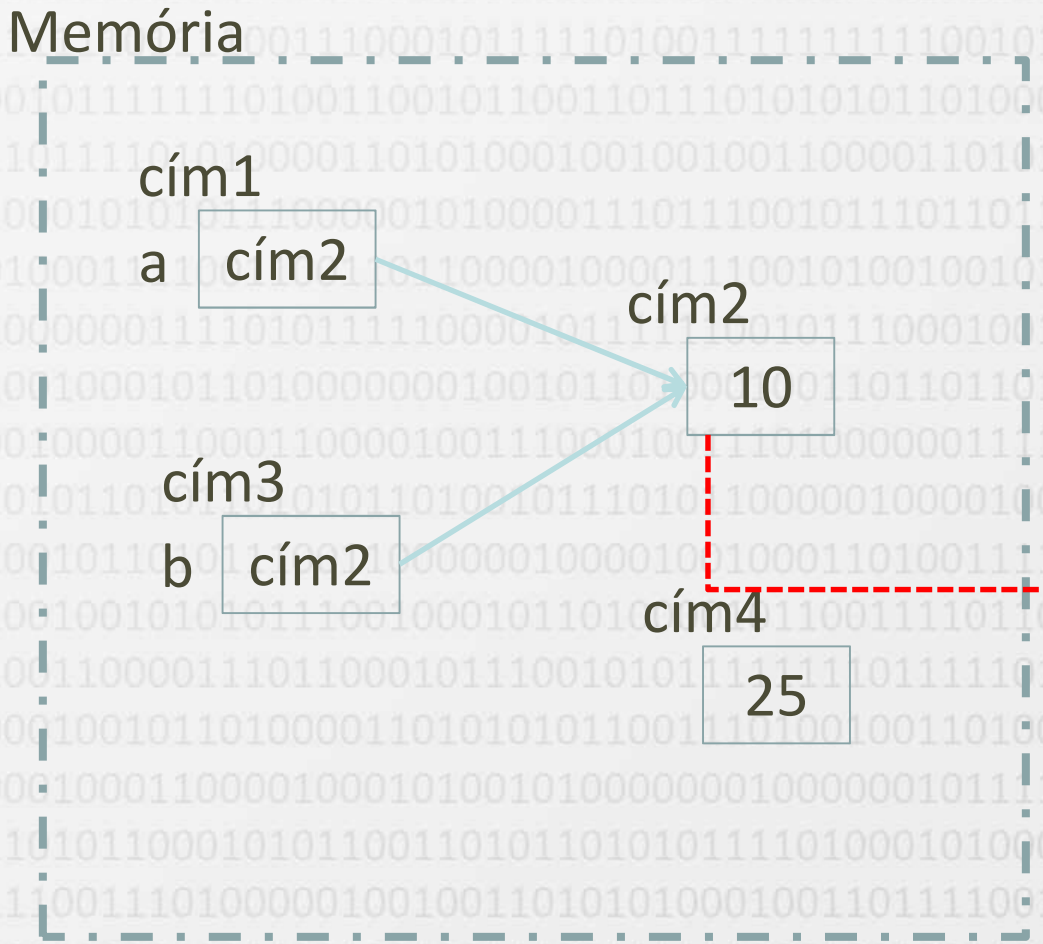
Referencia típusú változók értékadása!

Referencia típusú változók a memóriában 13. (sematikus ábra)



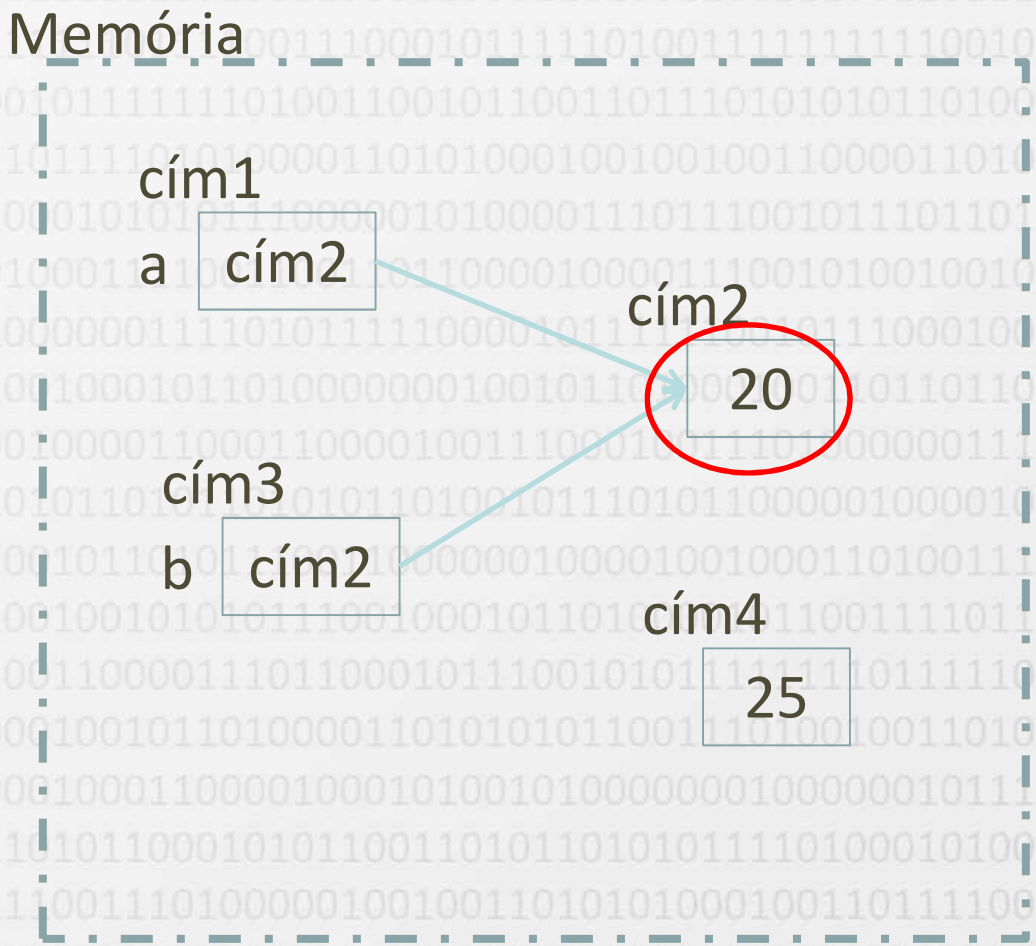

```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
Console.Write( a[0] );
```

Referencia típusú változók a memóriában 14. (sematikus ábra)



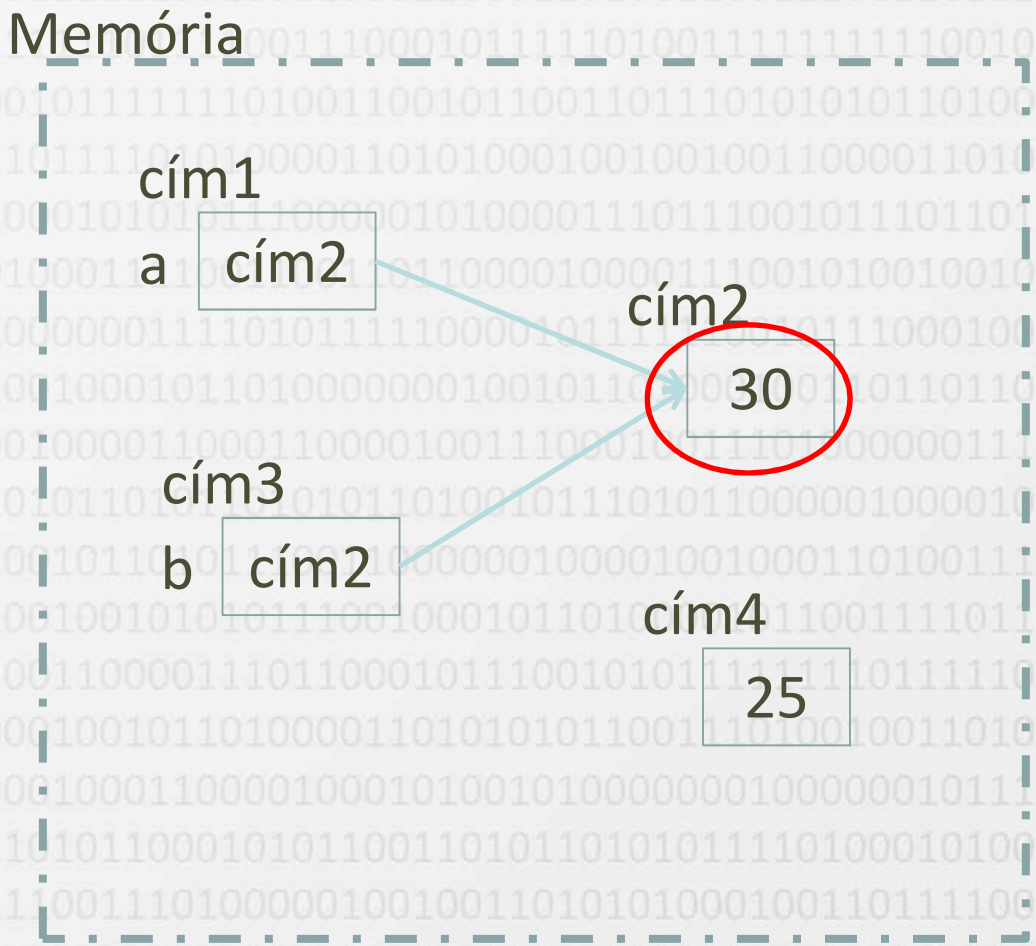

```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
Console.Write( a[0] );
Console.Write( b[0] );
```

Referencia típusú változók a memóriában 15. (sematikus ábra)



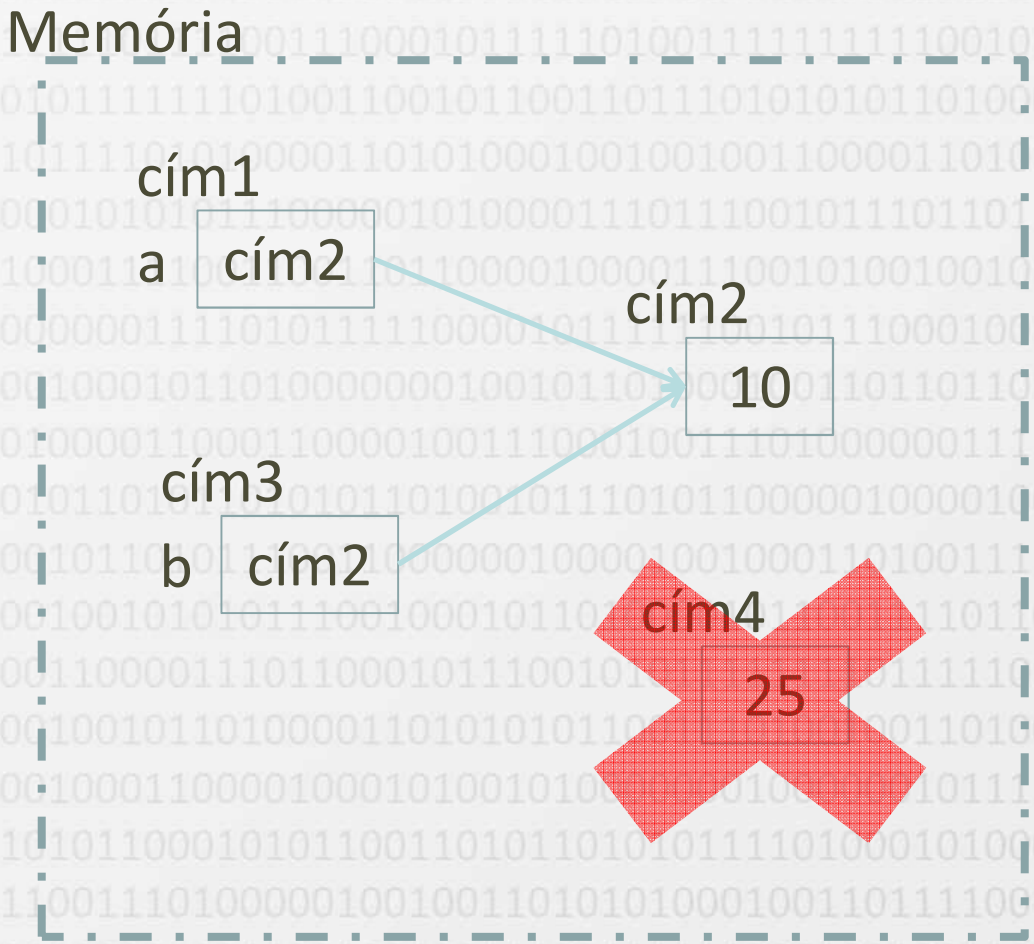

```
a[0] = 10;
Console.Write( a[0] );
Console.Write( b[0] );
b = a;
Console.Write( a[0] );
Console.Write( b[0] );
a[0] = 20;
```

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

Objektumorientált Programozás V.

A Microsoft Visual Studio 2010 használata
Műveletek tömbökkel
Érték- és referenciatípusú változók
Feladatok

Gyakorló feladatok

Olvasson be maximum n db. egész számot egy tömbbe, ezután döntse el a negatívok összegét, a pozitívok átlagát és a zérusok darabszámát!

Töltsön fel egy mátrixot FOR ciklusok segítségével, majd állítsa elő a transzponáltját (a sorokat fel kell cserélni az oszlopokkal), és írassa ki mindkét mátrixot!

Olvasson be egy mondatot, majd vizsgálja meg, hogy palindróma-e (tükörmondat, előlről és hátulról olvasva ugyanaz). A vizsgálat eredményét jelenítse meg. Palindrómák például: Géza kék az ég. Indul a görög aludni.

Objektumorientált Programozás V.

- ✓A Microsoft Visual Studio 2010 használata
- ✓Műveletek tömbökkel
- ✓Érték- és referenciatípusú változók
- ✓Feladatok

Irodalom, feladatok

- **Kotsis-Légrádi-Nagy-Szénási: Többnyelvű programozástechnika, PANEM, Budapest, 2007**
- **Faraz Rasheed: C# School, Synchron Data, 2006 http://www.programmersheaven.com/2/CSharpBook**
- **Reiter István: C# jegyzet, DevPortal, 2010, http://devportal.hu/content/CSharpjegyzet.aspx**