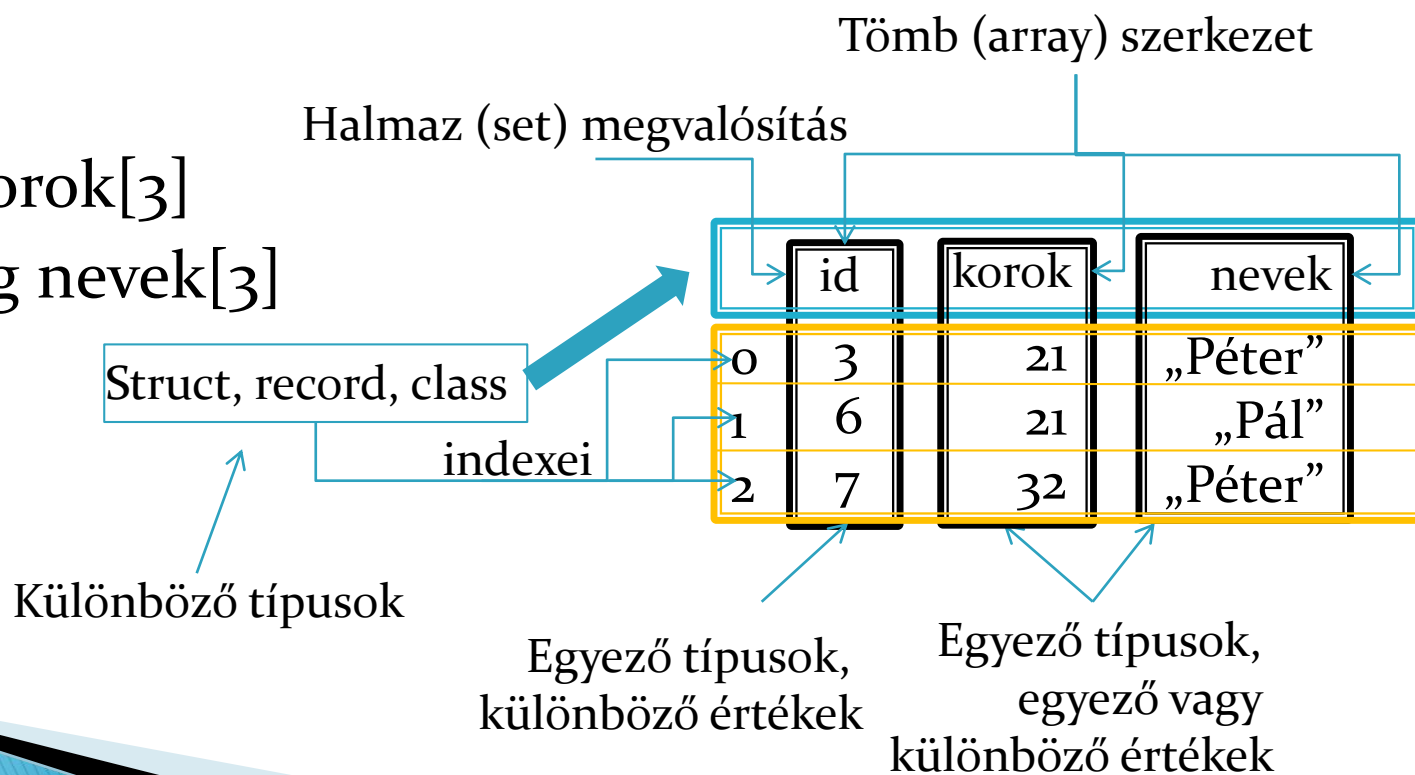


Összetett adatszerkezetek

- ▶ `int kor1, kor2, kor3;`
- ▶ `string nev1, nev2, nev3;`

Helyett:

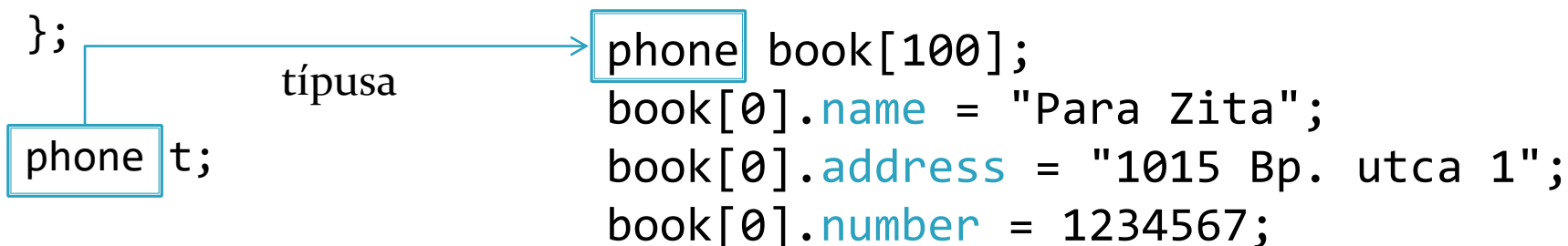
- ▶ `int korok[3]`
- ▶ `string nevek[3]`



Struktúra tömb

*Struct az class, long int az long javaban / C#-ban
Minden indexen a típusnak megfelelő adattagokat
érem el*

```
int main(){  
    struct phone {  
        string name;  
        string address;  
        long int number;  
    };
```

A diagram with a box labeled 'phone' next to the variable 't' in the code. An arrow points from this box to another box labeled 'phone' next to the array declaration 'book[100];'. The word 'típusa' (type) is written above the arrow.

```
    phone t;  
    phone book[100];  
    book[0].name = "Para Zita";  
    book[0].address = "1015 Bp. utca 1";  
    book[0].number = 1234567;
```

```
    t.name = "Kasza Blanka";  
    t.address = "1045 Erdőfásor u. 14.";  
    t.number = 1234567;
```

Tort [] használata – érték és referencia

```
Tort t0 = new Tort();
t0.nevezo = 3;
t0.szamlalo = 1;
```

```
Tort[] tortTomb = new Tort[2];
tortTomb[0] = t0;
tortTomb[1] = new Tort();
```

```
tortTomb[1].nevezo = 4;
tortTomb[1].szamlalo = 3;
```

- ▶ Milyen állapotban vannak a törtek a tömbben?

▲ t0	{TortProgram.Tort}
nevezo	3
szamlalo	1
▲ tortTomb	{TortProgram.Tort[2]}
▲ [0]	{TortProgram.Tort}
nevezo	3
szamlalo	1
▲ [1]	{TortProgram.Tort}
nevezo	0
szamlalo	0

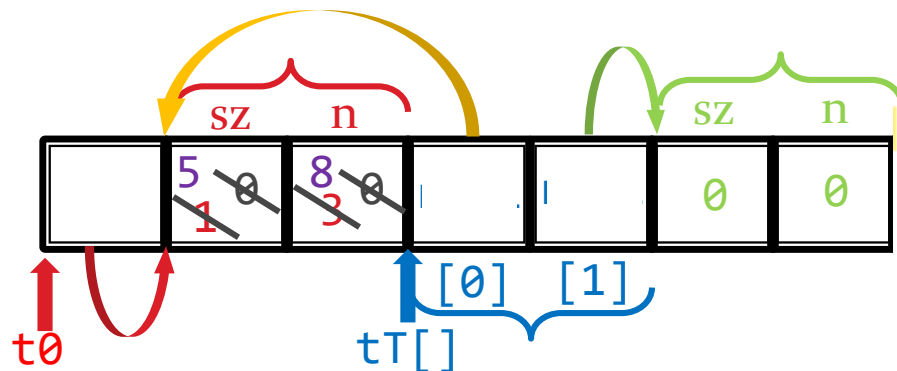
- ▶ A változtatások után?
- ▶ Miért?

▲ t0	{TortProgram.Tort}
nevezo	3
szamlalo	1
▲ tortTomb	{TortProgram.Tort[2]}
▲ [0]	{TortProgram.Tort}
nevezo	3
szamlalo	1
▲ [1]	{TortProgram.Tort}
nevezo	4
szamlalo	3

```
t0.nevezo = 8;
t0.szamlalo = 5;
```

▲ t0	{TortProgram.Tort}
nevezo	8
szamlalo	5
▲ tortTomb	{TortProgram.Tort[2]}
▲ [0]	{TortProgram.Tort}
nevezo	8
szamlalo	5
▲ [1]	{TortProgram.Tort}
nevezo	4
szamlalo	3

Tort [] a memóriában



A **t0** referencia
 ua. memóriaterületre hivatkozik,
 mint **tortTomb[0]** referenciája

```
public class Tort {
    int szamlalo; //SZ
    int nevezo; //n
}
```

```
Tort[] tortTomb = new Tort[2];
tortTomb[0].nevezo = 7;
```

Exception Thrown

System.NullReferenceException: 'Az objektumhivatkozás nincs beállítva semmilyen objektumpéldányra.'

```
Tort t0;
t0 = new Tort();
t0.nevezo = 3;
t0.szamlalo = 1;
```



```
Tort[] tortTomb = new Tort[2];
tortTomb[0] = t0;
tortTomb[1] = new Tort();
```

```
t0.nevezo = 8;
t0.szamlalo = 5;
```



Egységbezárás

unsigned: előjel nélküli, C#: uint
Java: minden előjeles (kivéve a char)

típus

```
struct Point{
    unsigned int x, y;
    int setX(unsigned int value){
        if(value <= MAX_X){
            x = value;
            return 0;
        }
        return -1;
    }
    int setY(unsigned int value){
        if(value <= MAX_Y){
            y = value;
            return 0;
        }
        return -1;
    }
};
```

azonosító

```
int main(){
    Point p1; //példányosítás
    p1.setX(100); //értékadás
    p1.setY(100); //értékadás

    //kiírások
    cout<<"p1.x: "<<p1.x<<endl;
    cout<<"p1.y: "<<p1.y<< endl;

    Point p2;
    p2.setX(200);
    p2.setY(200);

    cout<<"p2.x: "<<p2.x<<endl;
    cout<<"p2.y: "<<p2.y<<endl;
}
```

Adatrejtés – private láthatóság

- ▶ Az objektum elrejt bizonyos adatait, műveleteit.
- ▶ Így csak ellenőrzötten férhetünk hozzá, az állapotát (adattagok), csak saját műveletein (tagfüggvények) keresztül változtathatjuk.
- ▶ Az elrejtett adatokhoz csak az Osztály kódjában férünk hozzá.
- ▶ Az adatrejtés kulcsszava a láthatóság - Visibility

Láthatóság

Láthatósági
kulcsszavak →

```
struct Point{  
    private:  
        unsigned int x, y;  
    public:  
        int setX(unsigned int value);  
        int setY(unsigned int value);  
};
```

```
int main(){
```

```
    Point p1;
```

```
    p1.setX(100);  
    p1.setY(100);
```

← Adattagokat helyes, biztonságos elérése

```
    p1.x = 8888888;  
    p1.y = MAX_Y+3;
```

← Közvetlen elérjük az adattagokat!

← **private** -ként már nem fordul le!

Egységbezárás / adatrejtés

- ▶ **Private** adattagokon **public** metódusok dolgoznak:
 - Lekérdezik: **get**, boolean típusnál: **is**
 - Beállítják: **set**
- ▶ Írhatunk metódust saját elnevezéssel is, pl **terulet()**, **kerulet()**:
 - Ezek valójában lekérdezők, csak nem közvetlen az adattagot adják vissza, de velük végez műveletet


```
public class Auto { /* egységbezárás */  
    /* Adattagok: */  
    private String szin; /* adatrejtés */  
    private double uzemanyagSzint; /* adatrejtés */  
    private boolean voltTolve; /* adatrejtés */  
  
    /* publikus lekérdező és beállító tagfüggvények az adattagokra */  
    public String getSzin() { return szin; }  
    public void setSzin(String szin) { this.szin = szin; }  
  
    public double getUzemanyagSzint() { return uzemanyagSzint; }  
    public void setUzemanyagSzint(double uzemanyagSzint)  
    { this.uzemanyagSzint = uzemanyagSzint; }  
  
    public boolean isVoltTolve() { return voltTolve; }  
    public void setVoltTolve(boolean voltTolve)  
    { this.voltTolve = voltTolve; }  
}
```

Teglalap program

A téglalapnak nem lehet 1-nél kisebb oldala

- ▶ Létrehozzuk:
`Tegla t = new TegLa();`
- ▶ Kiírjuk:
`t.ki() -> a=0, b=0`

Teglalap
a, b: int
ki(): void

Konstruktor - Constructor

- ▶ Feladata, hogy megkonstruálja az objektumot (a memóriában).
- ▶ Az objektum adatait inicializálja

`Tegla t = new TegLa();`

- ▶ Nem írunk visszatérési értéket (void se), ua. a neve, mint az **Osztálynak**
- ▶ De van visszatérési értéke: referencia (**t**) az **objektumra** *(igazából a new intézi...)*
- ▶ Túl lehet terhelni
- ▶ Kaphat paramétert
- ▶ A kstr. explicit hívhat másik kstr-t, vagy implicite meghívódik

Konstruktor

```
int main( ){
    Line line = new Line();//konstr. meghívódik
    //itt a length az 0!!!
    line.setLength(6.0);
    cout<<„vonal hossza: „ << line.getLength();
}

class Line{
    public:
        void setLength(double len);
        double getLength(void);
        Line(); //constructor
    private:
        double length;
};

Line::Line( ){ cout << „Az obj. létrejött"; }

void Line::setLength( double len ){ length = len; }

double Line::getLength( void ){ return length; }
```

Konstruktor paraméterrel

```
int main( ){
    Line line = new Line(10.0); //konstr. meghívódik
    cout<<„vonal hossza: „ << line.getLength();

class Line{
    public:
        line.setLength(6.0);
        cout<<„vonal hossza: „ << line.getLength();

        void setLength(double len);
        double getLength(void);
        Line( double len );
    private:
        double length;
};

Line::Line(double len ){
    cout << „Az obj. létrejött";
    length = len;
}
```

Feladatok

- ▶ MindentTudo... – fájlkezeléssel
- ▶ VonalProgram – közösen
- ▶ EtteremProgram – OKJ
- ▶ Királynők – OKJ <http://infojegyzet.hu/vizsgafeladatok/>

C#

```
*** Vonalak lista elemei ***  
Vonal{kezd=0, hossz=5, tipus=DUPLA, szín=Yellow}  
====(5)  
Vonal{kezd=0, hossz=10, tipus=SZIMPLA, szín=White}  
----- (10)  
Vonal{kezd=5, hossz=10, tipus=SZIMPLA, szín=White}  
----- (10)  
Vonal{kezd=5, hossz=15, tipus=SZIMPLA, szín=Red}  
----- (15)  
Vonal{kezd=0, hossz=5, tipus=SZAGGATOTT, szín=Blue}  
..... (5)  
Vonal{kezd=0, hossz=5, tipus=SZIMPLA, szín=White}  
----- (5)
```