

typ záznam (record)

další způsob strukturování dat

```
record <seznam položek> end
```

```
type
```

```
  Komplex = record  
              Re, Im: real  
            end
```

```
var
```

```
  K: Komplex;  
  . . .  
  K.Re := 1.00;  
  K.Im := -5.25;
```

Dynamické proměnné

Strukturované příkazy

jednoduchý příkaz

složený příkaz

if, case

for-cyklus

while/repeat cyklus

procedura

Nevolá sama sebe

Volá sama sebe („rekurzivní“)

Strukturovaná data

skalární proměnná

záznam

~~záznam s variantami~~

pole

soubor

?

= nezajímavá.

=> data obsahující sama sebe!

Příklad

type

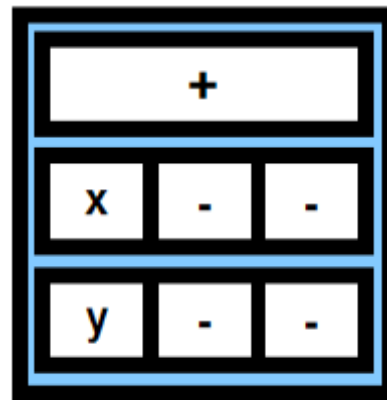
```
TVyraz = record
```

```
  operator: char; { +, -, *, / nebo promenna }
```

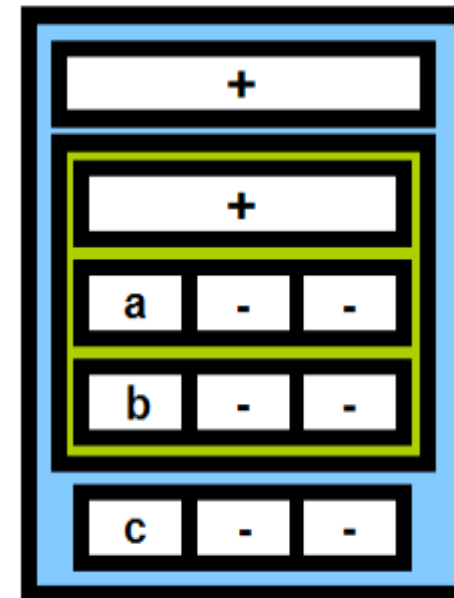
```
  pv1, pv2: TVyraz
```

```
end;
```

$x+y$

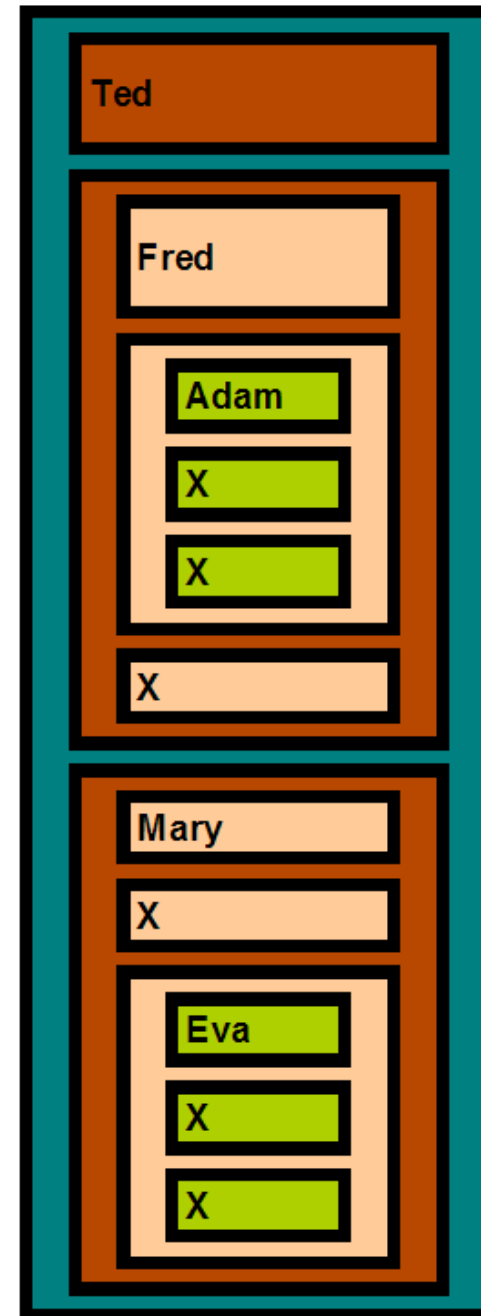


$a+b+c$



Příklad

```
type
  TRodokmen = record
    jmeno: string;
    { X znamena neznamy }
    otec, matka: TRodokmen
  end;
```



Zápis rekursivních dat

Ted

 Fred

 Adam

 X

 X

 X

 Mary

 X

 Adam

 X

 X

pomocí závorek:

((Ted, (Fred, (Adam, X, X) , X) , (Mary, X, (Eva, X, X))))

Typická vlastnost prvku:

VARIANTNOST

rekursivní procedura bez možnosti větvení
taky nikdy neskončí

PROMĚNNÁ VELIKOST

Důsledek:

Takovým proměnným NELZE při překladu
přiřadit pevný počet paměťových míst.

Důsledek důsledku:

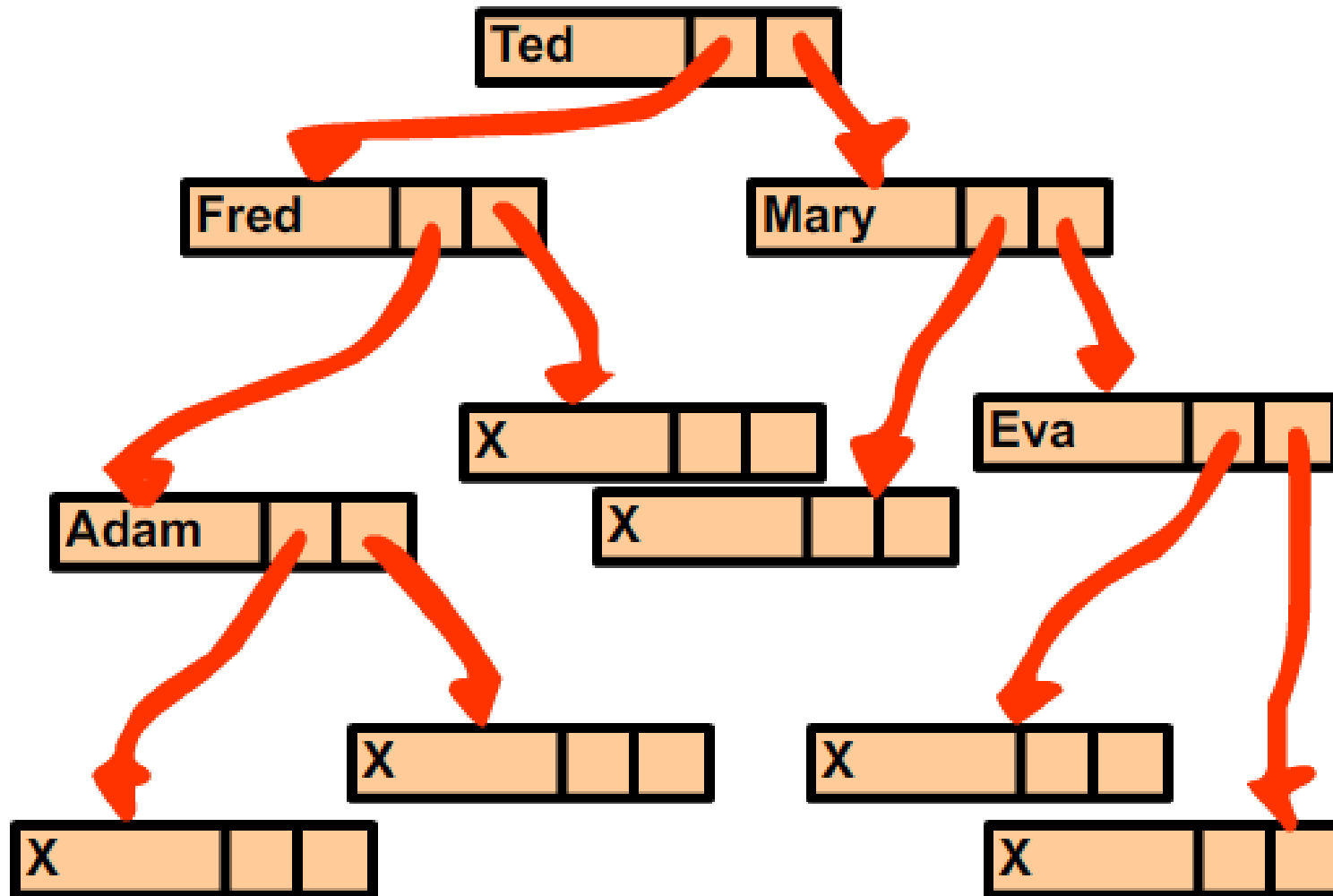
Kompilátor nemůže adresovat jednotlivé složky
takových rekursivních proměnných,
protože neví, KDE (a JESTLI VŮBEC) budou.

Jak se to tedy dělá?

- * paměť se přiděluje dynamicky jednotlivým složkám
- * kompilátor místo paměti pro další složku přidělí jen místo pro její ADRESU.

ADRESA = UKAZATEL = POINTER = SMERNÍK
= מצביע = نقطة = □□□□ = ...

Příklad



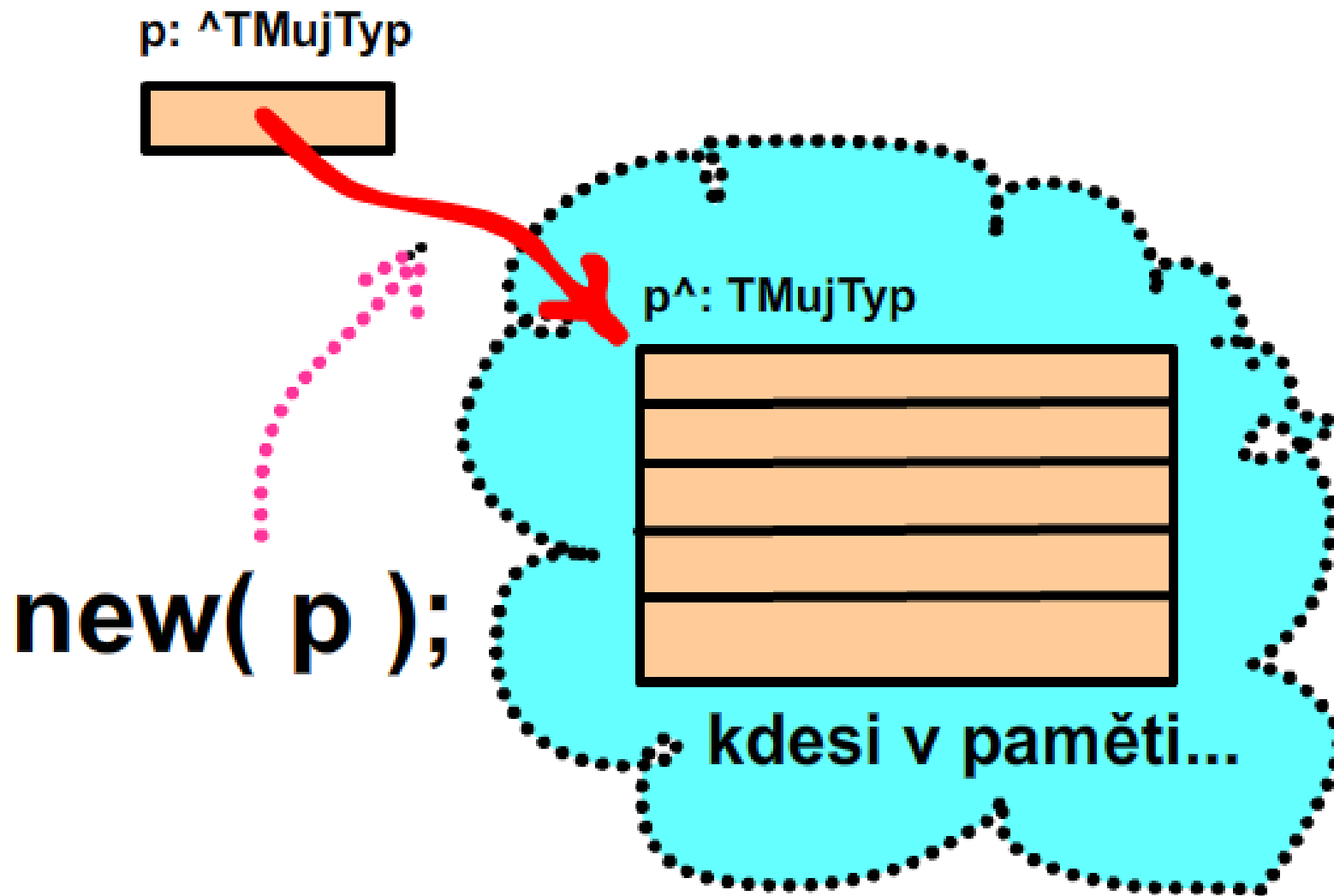
Jak je to v Pascalu

typ „ukazatel na typ TMujTyp“

```
type  
    PMujTyp = ^TMujTyp;
```

Ukazatel získá hodnotu, když vznikne (dostane paměť) prvek,
na který má ukazovat
= voláním procedury new

```
var  
    p: PMujTyp; { stejne jako p: ^TMujTyp }  
    ...  
    new( p );
```



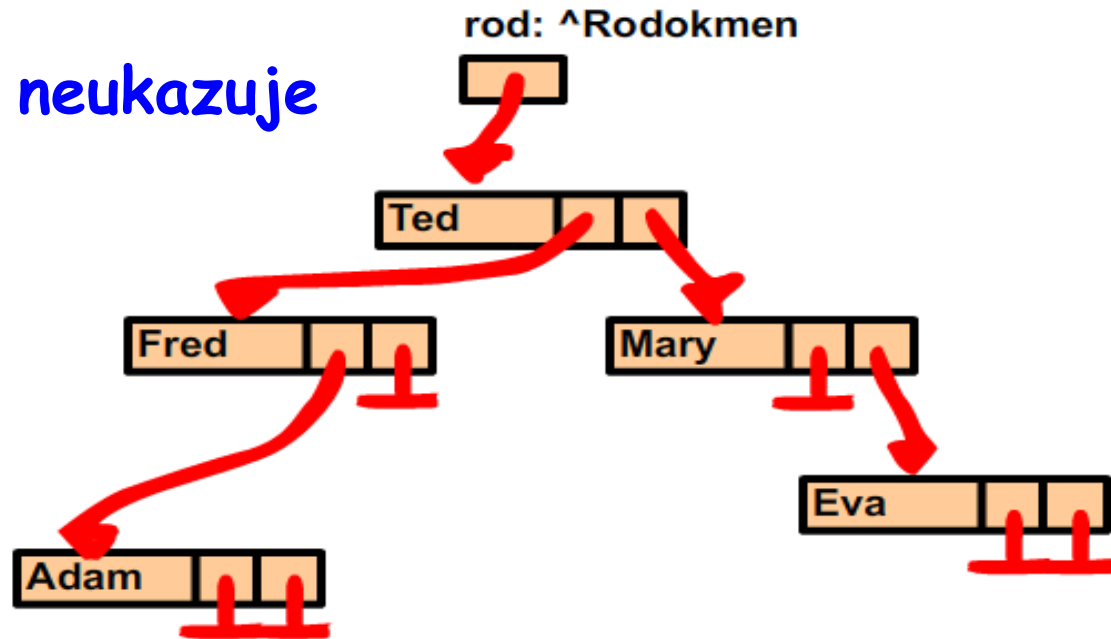
!!! NEW vytváří proměnné, které nemají vlastní jméno !!!

Ukazování ukazatelem

```
var rod: ^TRodokmen;  
...  
  new( rod ); { ted rod ukazuje na (novou)  
                promennou typu TRodokmen:  
                rod^ }  
rod^.Jmeno := 'Ted';  
rod^.otec  := ...
```

Konstanta NIL

ukazatel, který nikam neukazuje



```
new( rod );  
rod^.Jmeno := 'Ted';  
new( rod^.Otec );  
rod^.Otec^.Jmeno := 'Fred';  
new( rod^.Otec^.Otec );  
rod^.Otec^.Matka := NIL;  
...
```

Zrušení dynamické proměnné

```
dispose ( rod ) ;
```

Uvolní (vrátí) obsazenou paměť.

POZOR: Nedosazuje do té proměnné !

V C# automatická správa paměti (garbage/collector).

? Kam ukazuje NIL ?

Spojový seznam (jednosměrný, lineární)

type

```
  PPrvek = ^TPrvek;
```

```
  TPrvek = record
```

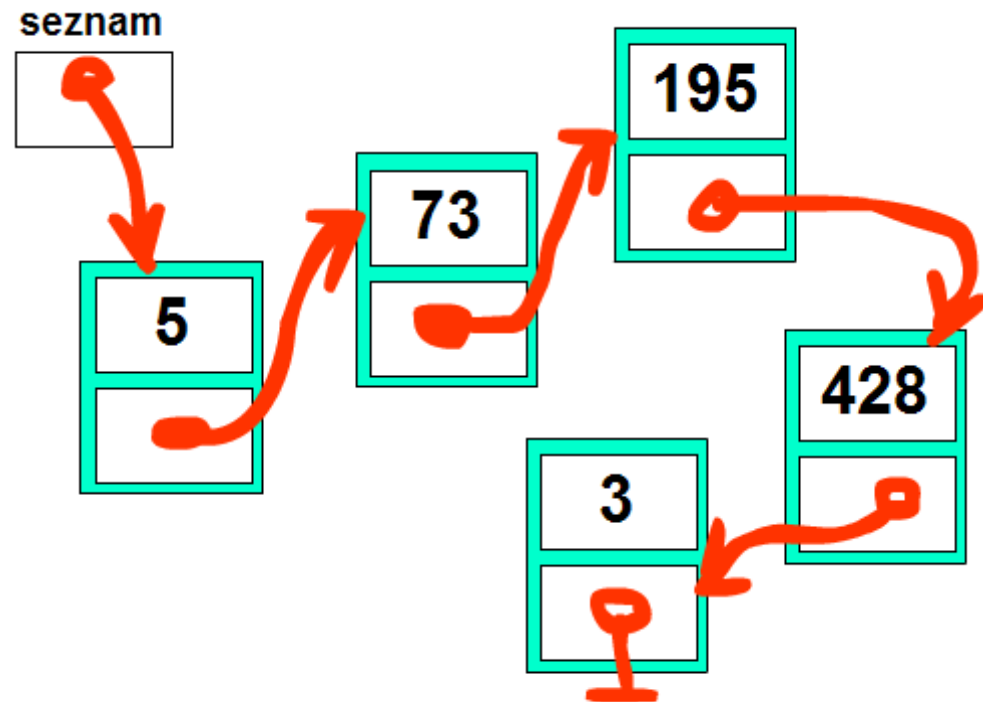
```
    Hodnota: NejakýTyp;
```

```
    Dalsi: PPrvek
```

```
end;
```

var

```
  seznam: PPrvek;
```



Základní kroky

projít

přidat na začátek

přidat na konec

zrušit první

zrušit poslední

zrušit obecně...

