

# Konkrétní datové typy – opakování a rozšíření

## Programovací techniky

doc. Ing. Jiří Rybicka, Dr.  
ústav informatiky  
PEF MENDELU v Brně  
`rybicka@mendelu.cz`



- Specifikace: množina povolených hodnot a množina povolených operací

- Specifikace: množina povolených hodnot a množina povolených operací
- Konkrétní datový typ: typ implementovaný v daném jazyce

- Specifikace: množina povolených hodnot a množina povolených operací
- Konkrétní datový typ: typ implementovaný v daném jazyce
- Každý jazyk má jiný repertoár konkrétních typů

- Specifikace: množina povolených hodnot a množina povolených operací
- Konkrétní datový typ: typ implementovaný v daném jazyce
- Každý jazyk má jiný repertoár konkrétních typů
- Datové typy pro stejný účel jsou v různých jazycích různě implementovány

- Specifikace: množina povolených hodnot a množina povolených operací
- Konkrétní datový typ: typ implementovaný v daném jazyce
- Každý jazyk má jiný repertoár konkrétních typů
- Datové typy pro stejný účel jsou v různých jazycích různě implementovány
- Jazyk Pascal: Datové typy jednoduché, strukturované, dynamické, objektové.





- V jednom okamžiku uchovávají jednu hodnotu

- V jednom okamžiku uchovávají jednu hodnotu
- Základní členění: ordinální, neordinální

- V jednom okamžiku uchovávají jednu hodnotu
- Základní členění: ordinální, neordinální
- Ordinální typy: jejich hodnoty jsou zobrazeny na vhodnou podmnožinu celých čísel (implementace)

- V jednom okamžiku uchovávají jednu hodnotu
- Základní členění: ordinální, neordinální
- Ordinální typy: jejich hodnoty jsou zobrazeny na vhodnou podmnožinu celých čísel (implementace)
- Neordinální typy: racionální čísla (zobrazeny pomocí dvojic celých čísel – mantisa, exponent)

- V jednom okamžiku uchovávají jednu hodnotu
- Základní členění: ordinální, neordinální
- Ordinální typy: jejich hodnoty jsou zobrazeny na vhodnou podmnožinu celých čísel (implementace)
- Neordinální typy: racionální čísla (zobrazeny pomocí dvojic celých čísel – mantisa, exponent)
- Ordinální typy: celá čísla (integer – základní ve všech verzích, rozšiřující jsou shortint, smallint, longint, byte, word, longword, cardinal); boolean, char, interval, výčet.



- Operace vyplývají ze zobrazení do množiny celých čísel

- Operace vyplývají ze zobrazení do množiny celých čísel
- Základní:  $\text{succ}(X)$ ,  $\text{pred}(X)$ , porovnání (6 operátorů)



- Operace vyplývají ze zobrazení do množiny celých čísel
- Základní:  $\text{succ}(X)$ ,  $\text{pred}(X)$ , porovnání (6 operátorů)
- Konverzní:  $\text{ord}(X)$ ,  $\text{typ}(I)$

- Operace vyplývají ze zobrazení do množiny celých čísel
- Základní:  $\text{succ}(X)$ ,  $\text{pred}(X)$ , porovnání (6 operátorů)
- Konverzní:  $\text{ord}(X)$ ,  $\text{typ}(I)$
- Rozšíření:  $\text{Inc}(X, [K])$ ,  $\text{Dec}(X, [K])$



- Celá čísla: aritmetické vlastnosti, doplňkový kód

- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota

- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota
- Rozšíření operátorů **and**, **or**, **not**.

- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota
- Rozšíření operátorů **and**, **or**, **not**.
- Příklady:  
    5 **or** 9 = 13    **not** 255 = 0  
    5 **and** 9 = 1    5 **xor** 9 = 12

- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota
- Rozšíření operátorů **and**, **or**, **not**.
- Příklady:  
    5 **or** 9 = 13    **not** 255 = 0  
    5 **and** 9 = 1    5 **xor** 9 = 12
- Bitové posuvy: **shr**, **shl**



- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota
- Rozšíření operátorů **and**, **or**, **not**.
- Příklady:  
    5 **or** 9 = 13    **not** 255 = 0  
    5 **and** 9 = 1    5 **xor** 9 = 12
- Bitové posuvy: **shr**, **shl**
- Posuv vlevo – do nejnižšího bitu se vloží vždy nula

- Celá čísla: aritmetické vlastnosti, doplňkový kód
- Logické vlastnosti: každý bit jako jedna logická hodnota
- Rozšíření operátorů **and**, **or**, **not**.
- Příklady:  
    5 **or** 9 = 13    **not** 255 = 0  
    5 **and** 9 = 1    5 **xor** 9 = 12
- Bitové posuvy: **shr**, **shl**
- Posuv vlevo – do nejnižšího bitu se vloží vždy nula
- Posuv vpravo – dva typy:  
    a) do nejvyššího bitu se vloží nula = **logický posuv**  
    b) do nejvyššího bitu se zopakuje původní hodnota = **aritmetický posuv**

# Využití bitových operací

- Umístění více logických informací do minimálního prostoru  
1001 = archivní, ne systémový, ne read only, adresář

- Umístění více logických informací do minimálního prostoru  
1001 = archivní, ne systémový, ne read only, adresář
- Maskování  
x **and** 3 – výběr nejnižších dvou bitů

- Umístění více logických informací do minimálního prostoru  
1001 = archivní, ne systémový, ne read only, adresář
- Maskování  
 $x \text{ and } 3$  – výběr nejnižších dvou bitů
- Zjištění hodnot vybraných bitů – příklady: zabezpečení paritou, šifrování  
kombinace  $\text{odd}(X)$  a posuvu vpravo

- Umístění více logických informací do minimálního prostoru  
1001 = archivní, ne systémový, ne read only, adresář
- Maskování  
 $x \text{ and } 3$  – výběr nejnižších dvou bitů
- Zjištění hodnot vybraných bitů – příklady: zabezpečení paritou, šifrování  
kombinace  $\text{odd}(x)$  a posuvu vpravo
- Násobení a dělení hodnotou se základem 2  
násobení nahrazeno posuvem doleva, dělení posuvem doprava





- Znakové hodnoty v jednobytovém kódování

- Znakové hodnoty v jednobytovém kódování
- Zápis hodnot pomocí obrazů v apostrofech nebo pomocí ordinálního čísla, např. #10

- Znakové hodnoty v jednobytovém kódování
- Zápis hodnot pomocí obrazů v apostrofech nebo pomocí ordinálního čísla, např. #10
- Operace: ordinální + UpCase

- Znakové hodnoty v jednobytovém kódování
- Zápis hodnot pomocí obrazů v apostrofech nebo pomocí ordinálního čísla, např. `#10`
- Operace: ordinální + UpCase
- Historická funkce `chr`

- Znakové hodnoty v jednobytovém kódování
- Zápis hodnot pomocí obrazů v apostrofech nebo pomocí ordinálního čísla, např. #10
- Operace: ordinální + UpCase
- Historická funkce `chr`
- Znakový kód ASCII:

řídící znaky	000x xxxx,
mezera	0010 0000
čísllice	0011 xxxx,
velká písmena	010x xxxx,
malá písmena	011x xxxx,
národní znaky	1xxx xxxx



- Výčet: Hodnotami jsou identifikátory

- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou



- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou
- Identifikátory představují pojmenované konstanty

- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou
- Identifikátory představují pojmenované konstanty
- Hlavní výhodou je zvýšení čitelnosti zdrojového textu

- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou
- Identifikátory představují pojmenované konstanty
- Hlavní výhodou je zvýšení čitelnosti zdrojového textu
- Nutnost uživatelských konverzí při textovém vstupu a výstupu

- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou
- Identifikátory představují pojmenované konstanty
- Hlavní výhodou je zvýšení čitelnosti zdrojového textu
- Nutnost uživatelských konverzí při textovém vstupu a výstupu
- Interval – zúžení hodnot a operací na podmnožinu jiného ordinálního typu

- Výčet: Hodnotami jsou identifikátory
- Implementace celými čísly počínaje nulou
- Identifikátory představují pojmenované konstanty
- Hlavní výhodou je zvýšení čitelnosti zdrojového textu
- Nutnost uživatelských konverzí při textovém vstupu a výstupu
- Interval – zúžení hodnot a operací na podmnožinu jiného ordinálního typu
- Určen ke kontrole rozsahů hodnot, při zapnutí kontrol překladače



- V jednom okamžiku uchovávají více hodnot

- V jednom okamžiku uchovávají více hodnot
- Lze rozdělit na homogenní (složky stejného typu) a heterogenní (složky různých typů)



- V jednom okamžiku uchovávají více hodnot
- Lze rozdělit na homogenní (složky stejného typu) a heterogenní (složky různých typů)
- **Bázový typ** = typ složky

- V jednom okamžiku uchovávají více hodnot
- Lze rozdělit na homogenní (složky stejného typu) a heterogenní (složky různých typů)
- **Bázový typ** = typ složky
- Základní operací je přístup ke složce

- V jednom okamžiku uchovávají více hodnot
- Lze rozdělit na homogenní (složky stejného typu) a heterogenní (složky různých typů)
- **Bázový typ** = typ složky
- Základní operací je přístup ke složce
- Paměťové typy lze přiřazovat

- V jednom okamžiku uchovávají více hodnot
- Lze rozdělit na homogenní (složky stejného typu) a heterogenní (složky různých typů)
- **Bázový typ** = typ složky
- Základní operací je přístup ke složce
- Paměťové typy lze přiřazovat
- Specifické možnosti: řetězec, množina, soubor



- Pole představuje nejbližší obraz operační paměti

- Pole představuje nejbližší obraz operační paměti
- Koncepce pole v jazyce Pascal – vždy statické, jednorozměrné

- Pole představuje nejbližší obraz operační paměti
- Koncepce pole v jazyce Pascal – vždy statické, jednorozměrné
- Složkou pole může být libovolný typ bez omezení



- Pole představuje nejbližší obraz operační paměti
- Koncepce pole v jazyce Pascal – vždy statické, jednorozměrné
- Složkou pole může být libovolný typ bez omezení
- Přístup ke složce pomocí indexu

- Pole představuje nejbližší obraz operační paměti
- Koncepce pole v jazyce Pascal – vždy statické, jednorozměrné
- Složkou pole může být libovolný typ bez omezení
- Přístup ke složce pomocí indexu
- Indexový výraz:

$$A = A_0 + (I - I_0) \cdot V,$$

kde  $A$  je výsledná adresa,  $A_0$  je počáteční adresa pole,  $I$  je aktuální index,  $I_0$  je index prvního prvku pole,  $V$  je velikost složky pole v bytech.

- Pole představuje nejbližší obraz operační paměti
- Koncepce pole v jazyce Pascal – vždy statické, jednorozměrné
- Složkou pole může být libovolný typ bez omezení
- Přístup ke složce pomocí indexu
- Indexový výraz:

$$A = A_0 + (I - I_0) \cdot V,$$

kde  $A$  je výsledná adresa,  $A_0$  je počáteční adresa pole,  $I$  je aktuální index,  $I_0$  je index prvního prvku pole,  $V$  je velikost složky pole v bytech.

- Optimalizace indexového výrazu:
  - a)  $I_0 = 0$ ,
  - b)  $V = 2^n$ , příp.  $V = 1$



- Implementován jako pole `array [0..L] of char`

- Implementován jako pole `array [0..L] of char`
- První složka představuje okamžitou délku

- Implementován jako pole `array [0..L] of char`
- První složka představuje okamžitou délku
- Indexy od 1 do L reprezentují jednotlivé znaky

- Implementován jako pole `array [0..L] of char`
- První složka představuje okamžitou délku
- Indexy od 1 do L reprezentují jednotlivé znaky
- Deklarovaná délka L je max. 255



- Implementován jako pole `array [0..L] of char`
- První složka představuje okamžitou délku
- Indexy od 1 do L reprezentují jednotlivé znaky
- Deklarovaná délka L je max. 255
- Odvozen od typu `char` – jednobytové kódování

- Implementován jako pole `array [0..L] of char`
- První složka představuje okamžitou délku
- Indexy od 1 do L reprezentují jednotlivé znaky
- Deklarovaná délka L je max. 255
- Odvozen od typu `char` – jednobytové kódování
- Řada operací: okamžitá délka, porovnání, hledání, vkládání a mazání podřetězce, zřetězení



- Jediný heterogenní typ

- Jediný heterogenní typ
- Široké použití pro reprezentaci dat (databáze, dynamické struktury)

- Jediný heterogenní typ
- Široké použití pro reprezentaci dat (databáze, dynamické struktury)
- Přístup ke složce prostřednictvím tečkové notace

- Jediný heterogenní typ
- Široké použití pro reprezentaci dat (databáze, dynamické struktury)
- Přístup ke složce prostřednictvím tečkové notace
- Usnadnění přístupu – příkaz `with`

- Jediný heterogenní typ
- Široké použití pro reprezentaci dat (databáze, dynamické struktury)
- Přístup ke složce prostřednictvím tečkové notace
- Usnadnění přístupu – příkaz `with`
- Variantní záznam – některé složky mohou sdílet stejné místo v paměti, používají se výlučně



- Jediný heterogenní typ
- Široké použití pro reprezentaci dat (databáze, dynamické struktury)
- Přístup ke složce prostřednictvím tečkové notace
- Usnadnění přístupu – příkaz `with`
- Variantní záznam – některé složky mohou sdílet stejné místo v paměti, používají se výlučně
- Operace se záznamem jako celkem nejsou (jen přiřazení identických záznamů)



- Specifický typ jazyka Pascal

- Specifický typ jazyka Pascal
- Implementována jako bitové pole

- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem

- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem
- Hodnota je bit v poloze 0 – nepřítomný prvek, 1 – přítomný prvek

- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem
- Hodnota je bit v poloze 0 – nepřítomný prvek, 1 – přítomný prvek
- Bázovým typem je ordinální typ s hodnotami v intervalu ordinálních čísel 0 až 255

- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem
- Hodnota je bit v poloze 0 – nepřítomný prvek, 1 – přítomný prvek
- Bázovým typem je ordinální typ s hodnotami v intervalu ordinálních čísel 0 až 255
- Free Pascal: dvě velikosti, 4B a 32B



- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem
- Hodnota je bit v poloze 0 – nepřítomný prvek, 1 – přítomný prvek
- Bázovým typem je ordinální typ s hodnotami v intervalu ordinálních čísel 0 až 255
- Free Pascal: dvě velikosti, 4B a 32B
- Operace – sjednocení, průnik, rozdíl, zjištění přítomnosti prvku

- Specifický typ jazyka Pascal
- Implementována jako bitové pole
- Prvek je indexem
- Hodnota je bit v poloze 0 – nepřítomný prvek, 1 – přítomný prvek
- Bázovým typem je ordinální typ s hodnotami v intervalu ordinálních čísel 0 až 255
- Free Pascal: dvě velikosti, 4B a 32B
- Operace – sjednocení, průnik, rozdíl, zjištění přítomnosti prvku
- Užitečná pomůcka při přetypování



- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti
- Rozdělení souborů – textové, netextové; operace jsou rozdílné

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti
- Rozdělení souborů – textové, netextové; operace jsou rozdílné
- Netextové soubory mohou nebo nemusí mít udán typ datové složky



- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti
- Rozdělení souborů – textové, netextové; operace jsou rozdílné
- Netextové soubory mohou nebo nemusí mít udán typ datové složky
- Textové soubory mají čistě sekvenční charakter

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti
- Rozdělení souborů – textové, netextové; operace jsou rozdílné
- Netextové soubory mohou nebo nemusí mít udán typ datové složky
- Textové soubory mají čistě sekvenční charakter
- Netextové soubory umožňují přímý přístup ke složkám, jde o obdobu pole

- Jediný datový typ, jehož proměnné nejsou uloženy v operační paměti
- Veškeré manipulace jsou řešeny podprogramy – jazykové rozhraní k systémovým službám
- Princip přístupu k souboru – buffer v paměti
- Rozdělení souborů – textové, netextové; operace jsou rozdílné
- Netextové soubory mohou nebo nemusí mít udán typ datové složky
- Textové soubory mají čistě sekvenční charakter
- Netextové soubory umožňují přímý přístup ke složkám, jde o obdobu pole
- Operace: Assign, reset, rewrite, read, readln, write, writeln, eof, eoln, Close, Seek, FilePos, FileSize