Praktikum z logického programování (kombi) - Cvičení č. 4

KOMBINOVANÉ: ZÁPOČET

ABSTRAKTNÍ DATOVÉ TYPY (ADT) A STRUKTURY V PROLOGU

Datový typ

- Určuje obor hodnot, kterých smí proměnná s daným datovým typem nabývat
- Definuje množinu operací nad danými hodnotami
- Základní datové typy:
  - Logická hodnota
  - Celé číslo
    - Znak
    - Výčet
  - Reálné číslo
- Složené datové typy

Složený datový typ

- Heterogenní vs. Homogenní struktura
- Závisí na implementaci v daném jazyce
  - Některé typy jsou vestavěné (pole v C vs. seznam v Prologu)
  - Ostatní využívají struktury vestavěných datových typů (ADT)
- Příklad:
  - Pole
  - Textový řetězec
  - Seznam
  - Záznam (strukturovaný typ)

Seznam v Prologu (opakování)

- Vestavěný datový typ
  - Přístup k čelu seznamu a zbytku seznamu [H|T]
- Základní operace:
  - Vložení na začátek nebo na konec seznamu
  - Vyhledání prvku (zisk nebo existence)
  - Odstranění prvku (první, všech, …)
- Další operace:
  - Spojení seznamů
  - Délka
  - …

Pole (vektor) v prologu

- Není vestavěným datovým typem → nutno implementovat pomocí seznamu
- Indexovaná struktura – přístup k prvku pomocí jeho indexu (celého čísla, označujícího pořadí prvku v rámci pole)
  - Indexace od nuly vs. Indexace od jedničky
- Základní operace:
  - Přístup k prvku pomocí indexu
    - Zisk prvku
    - Změna prvku
  - Vyhledání prvku – vrácení indexu

Pole – zisk prvku na indexu

- Přistupujeme k i-tému prvku seznamu
- ityPrvek/3: rekurzivně procházíme seznam a dekrementujeme čítač dokud není čítač rovný nule (jedné)
- ityPrvek([Prvek|_], 0, Prvek) :- !.
- ityPrvek([_|Telo], Index, Prvek) :-
  DalsiIndex is Index - 1,
  ityPrvek(Telo, DalsiIndex, Prvek).

Pole – přepis prvku na indexu

- Přistupujeme k i-tému prvku seznamu, tentokrát ovšem potřebujeme vrátit nový seznam se změněným i-tým prvkem
- Rekurzivně procházíme seznamem a odkládáme si vždy prvek z čela seznamu, jakmile dorazíme na daný index, zahodíme prvek z čela původního seznamu a na jeho místo dáme vkládaný prvke, zbytek seznamu neměníme.
- zmenItyPrvek(Prvek, 0, [_|Telo], [Prvek|Telo]) :- !.
- zmenItyPrvek(Prvek, Index, [H|T], [H|T1]) :-
  DalsiIndex is Index - 1,
  zmenItyPrvek(Prvek, DalsiIndex, T, T1).

Úkol: vyhledání prvku pole – vrácení indexu

- Implementujte pomocí seznamu v Prologu
- indexPrvkuPole(+Prvek, +Pole, -Index).

Abstraktní datové typy (ADT)

- Implementačně nezávislá specifikace struktury dat s operacemi povolenými na této struktuře
  - Jsou definovány operace a to jak mají nad daty pracovat, ale už se neřeší daná implementace, ta je na konkrétním jazyce či programátorovi
- Příklady:
  - Asociativní pole
  - Zásobník
  - Fronta
  - Seznam
  - Množina
  - Strom
  - …

ADT Zásobník (stack)

- Data uložena jako poslední budou čtena jako první
  - LIFO (Last In – First Out)
- Operace nad zásobníkem:
  - přidání položky na vrchol zásobníku (**push**)
  - odebrání položky z vrcholu zásobníku (**pop**)
  - zisk položky z vrcholu zásobníku bez odebrání (**top**)
  - dotaz na prázdnost zásobníku (**is_empty**)

Úkol: operace nad zásobníkem

- Implementujte pomocí seznamu v Prologu
- push(+Prvek, +Zasobnik, -NovyZasobnik).
- pop(-Prvek, +Zasobnik, -NovyZasobnik).
- top(-Prvek, +Zasobnik).
- is_empty(+Zasobnik).

ADT Fronta (pipe)

- Data uložena jako první budou čtena jako první
  - FIFO (First In – First Out)
- Operace nad frontou:
  - přidání položky na konec fronty (**enqueue**)
  - výběr položky ze začátku fronty (**dequeue**)
  - zisk položky ze začátku fronty bez jejího odebrání (**front**)
  - dotaz na prázdnost fronty (**is_empty**)

Úkol: operace nad frontou

- Implementujte pomocí seznamu v Prologu
- enqueue(+Prvek, +Fronta, -NovaFronta).
- dequeue(-Prvek, +Fronta, -NovaFronta).
- front(-Prvek, +Fronta).
- is_empty(+Fronta).

ADT Množina (set)

- Data uložena tak, že se **jednotlivé prvky neopakují** a **nezáleží na jejich pořadí**
- Operace nad množinou:
  - přidání položky do množiny (**add)**
  - odebrání položky z množiny (**remove**)
  - test na přítomnost položky v množině (**member**)
  - Logické operace:
    - sjednocení (**union**)
    - průnik (**intersection**)
    - rozdíl (**difference**)

Úkol: operace nad množinou

- Implementujte pomocí seznamu v Prologu
- union(+PrvniMnozina, +DruhaMnozina,
  -VyslednaMnozina).
- intersection(+PrvniMnozina, +DruhaMnozina,
  -VyslednaMnozina).
- difference(+PrvniMnozina, +DruhaMnozina,
  -VyslednaMnozina).