**BI-SPOL-22 Abstraktní datový typ, jeho specifikace a implementace. Zásobník, fronta, pole, seznam, tabulka, množina. Implementace pomocí pole, spojových struktur a stromů**

BI-PA2

Datové typy jsou užívány pro

* definování množiny hodnot
* definování množiny přípustných operací

Programovací jazyky mají řadu zabudovaných datových typů – ty mají svou implementaci (paměťovou reprezentaci a operace)

### Abstraktní datový typ

* ADT je jistá specifikace struktury dat, která definuje obecnou sémantiku chování (definuje množiny hodnot a operací daného datového typu), není závislá na konkrétní implementaci (může být implementována různě – efektivita, jazyky, apod.)
* ADT definují:
  + množinu hodnot,
  + množinu operací,

nezávisle na konkrétní implementaci

* ADT může být formálně specifikován signaturou operací (definice syntaxe) a množinou axiomů (definice sémantiky):
  + Axiomy jsou ekvivalence mezi výrazy, každý výraz reprezentuje stav (obsah) ADT
  + Výrazy jsou složeny z operací a proměnných
  + Axiomy mohou být použity pro zjednodušení komplexnějších výrazů

Obsah obrázku text

Popis byl vytvořen automatickyObsah obrázku text

Popis byl vytvořen automaticky**Formální (axiomatický) popis zásobníku** –ukázka

**Implementace**

* některé programovací jazyky (např. jazyk Clear) dovolují formální specifikaci ADT
* imperativní a OOP vyžadují explicitní implementaci ADT
* V C++ je doporučeno implementovat ADT generickými třídami
  + Typ prvku je generickým parametrem šablony třídy
  + Operace init je implementována konstruktorem
  + Implementace obvykle používá dynamicky alokovanou paměť, proto je často vyžadován i destruktor, kopírující konstruktor a přetížený operátor =
  + Když je signatura operace ADT, … -> elem, je doporučená implementace const metodou vracející T
  + Když je signatura operace ADT, … -> ADT, je doporučená implementace metodou modifikující objekt

### Struktury

#### Zásobník (stack)

**Specifikace:**

* princip LIFO (Last In, First Out)
* přistupovat se dá pouze k vrcholu zásobníku
* přidávat/odebírat také pouze na/z vrcholu
* Interface: init, push, pop, top (vrchol), empty
* Složitost: O(1)
* Dynamicky alokované pole – O(n) (když se mění velikost) (amortizovaně je to O(1))

**Implementace**:

1. pole pevné délky, kapacita (maximální počet prvků v zásobníku) omezena už při kompilaci statickou velikostí pole
2. dynamicky alokované pole, velikost pole je dána parametrem konstruktoru, opět omezená velikost
3. dynamicky alokované pole, velikost pole se mění vždy, když je třeba
4. Obsah obrázku text

   Popis byl vytvořen automatickyjednosměrně zřetězený spojový seznam

* časová složitost je konstantní jak pro metodu push, tak pro pop
* v případě dynamicky alokovaného pole s měnící se velikostí je časová složitost push lineární
* příklad implementace ->

#### Fronta (queue)

**Specifikace:**

* FIFO (First In, First Out)
* přistupovat se dá pouze k začátku, přidává se na konec
* Interface: init, empty, add, front, remove
* Složitost: O(1)
* Dynamicky alokované pole – O(n) (když se mění velikost) (amortizovaně je to O(1))

**Implementace:**

* 1. pole pevné délky, kapacita (maximální počet prvků v zásobníku) omezena už při kompilaci statickou velikostí pole
  2. dynamicky alokované pole, velikost pole je dána parametrem konstruktoru, opět omezená velikost
  3. dynamicky alokované pole, velikost pole se mění vždy, když je třeba
  4. Obsah obrázku text

     Popis byl vytvořen automatickyObsah obrázku text

     Popis byl vytvořen automatickyjednosměrně zřetězený spojový seznam

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text, osoba, snímek obrazovky, dokument

Popis byl vytvořen automaticky

#### Seznam (list)

**Specifikace**

* Datová struktura, která poskytuje operace *insert, remove, read*
* Operace jsou určeny pozicí v seznamu. Pozice může být měněna
* Složitost: O(1) (když chceme find, tak je to O(n))

**Implementace**

1. Zřetězený seznam (řetěz struktur) – umožňuje provádět vložení a rušení s konstantní časovou složitostí

* Nemůže být použito pole, protože chceme konstantní časovou složitost (vložení prvku doprostřed pole vyžaduje lineární čas pro posun prvků, totéž pro rušení

Obsah obrázku text

Popis byl vytvořen automaticky

#### Pole (array)

Specifikace:

* datový kontejner organizující prvky v *n*-dimenzionálním prostoru
  + náhodný přístup k prvkům s konstantní časovou složitostí
  + prvek je identifikován n-ticí indexů (celých čísel)
* Obsah obrázku text

  Popis byl vytvořen automatickySložitost: O(1) pro náhodný přístup
  + Zvětšování pole je kopírování – O(n)
* Popis pole:
  + Datový typ prvků
  + Počet dimenzí
  + Dolní a horní meze jednotlivých dimenzí

Implementace

* Jednorozměrné, vícerozměrné pole

**Jednodimenzionální pole**

Obsah obrázku text

Popis byl vytvořen automaticky

**Multidimenzionální pole**

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

#### Tabulka (map)

**Specifikace**

* kontejner obsahující dvojici klíč-hodnota, kde klíč je unikátní
* Obsah obrázku text

  Popis byl vytvořen automaticky
* Mapy a množiny jsou podobné. Množinu můžeme považovat za speciální případ mapy, kde hodnoty jsou typu bool
* Generická třída implementující mapu:

Obsah obrázku text

Popis byl vytvořen automaticky

**Implementace**

1. pole, kde klíče jsou indexy v poli (klíče jsou celočíselné)
2. pole (neseřazené)
3. pole (seřazené)
4. spojový seznam (jednosměrný, neseřazený)
5. spojový seznam (jednosměrný, seřazený)
6. binární vyhledávací strom
7. rozptylovací tabulka (hash table)

**Přímý přístup**

Obsah obrázku text

Popis byl vytvořen automaticky

**Neseřazené pole**

Obsah obrázku text, osoba, snímek obrazovky

Popis byl vytvořen automaticky

**Seřazené pole**

Obsah obrázku text

Popis byl vytvořen automaticky

#### Množina (set)

**Specifikace:**

* kontejner obsahující prvky typu T bez duplikátů
* operace: *insert*, *remove*, *hasElement* a množinové operace

**Implementace**:

1. indikátorový (charakteristický) vektor – pokud je universum dostatečně malé a konečné, udělá se bool vektor nebo bitové pole (0 - není, 1 - je), vše je pak časově konstantní
2. (ne)seřazené pole - vše O(n), kromě neseřazený add O(1) a seřazený find O(log n)
3. jednosměrný (ne)seřazený spojový seznam – vše O(n)
4. binární vyhledávací strom
5. hash tabulka

**Indikátorový vektor**

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

**Neseřazené pole**

* prvky jsou umístěny v poli
* prvky v poli nejsou seřazeny
* pole je alokováno dynamicky a jeho velikost je měněna podle potřeby
* třída musí mít přehled o velikosti pole a o počtu prvků v množině

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

**Seřazené pole**

* prvky jsou umístěny v poli, seřazeném v rostoucím (klesajícím) pořadí
* pole je alokováno dynamicky a jeho velikost je měněna podle potřeby
* třída musí mít přehled o velikosti pole a o počtu prvků v množině

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

**Spojový seznam**

Obsah obrázku text

Popis byl vytvořen automaticky

**Přehled**

Obsah obrázku text

Popis byl vytvořen automaticky

Otázky

* ADT množina, co je to ADT, typy implementací, časová a paměťová složitost operací:
* Datový typ tabulka: implementace, časová a paměťová složitost.
* ADT - Set & List, rozdiely, implementácia