Abstrakte Datentypen
Datenstrukturen

Lernziele

- Sie kennen das Konzept der abstrakten Datentypen
- Sie kennen den Unterschied zwischen ADT und Datenstruktur

Abstrakte Datentypen

- Konzept welches einen Datentyp logisch definiert
- Datenstruktur und Operationen, die auf der Datenstruktur ausgeführt werden können
- Es werden keine konkreten Implementierungen der Operationen definiert
- Definiert eine «Idee» → Konzept
  - Legt fest, welche Operationen was tun (Semantik)
  - Aber nicht wie (konkrete Implementierung)
  - Kapselung durch Definition einer Schnittstelle
- Hat nichts zu tun mit abstrakten Klassen (C#)

Wichtig bei Datentypen …

- … ist, was sie tun (= Schnittstelle)
- … ist nicht, wie sie es tun (= Implementierung)

Abstrakte Datentypen

- Abstraktion
  - ADT abstrahieren von Implementierung.
  - Wird in der Informatik häufig verwendet:
    - ⇨„Bit“ abstrahiert von physikalischen Zuständen.
    - ⇨CPU abstrahiert von einer Menge von Bits und deren Zustandsänderungen.
    - ⇨Java Virtual Machine abstrahiert von konkreter CPU.

ADT – Definition (1)

Datentyp

- Die in einer Programmiersprache vorhandenen
  - Grundtypen, wie int, boolean, real, char, …, und
  - Strukturierungsmethoden, wie array, struct, …

  und die darauf definierten Operationen wie
  + (Addition, Stringkonkatenation), *, sqrt, [] (Selektion), …

Anmerkung:

- Datentypen hängen von der Programmiersprache ab.
- Es gibt heute ein Verständnis über „übliche“ Datentypen:
  - ganze Zahlen, Fließkommazahlen, Zeichen, Zeichenketten.
  - Gruppierung gleichartiger und verschiedener Datentypen.

ADT – Definition (2)

ADT (abstrakter Datentyp)

- Ein Datentyp, d.h.
  - eine **Menge** von **Werten** und
  - **Operationen** auf diesen Werten,
  - der **nur** über eine **Schnittstelle** zugänglich ist, sowie
  - **Regeln** über die Wirkung der Operationen auf den Werten.

Implementierung (eines ADT)

- Ein Programm, das den Datentyp realisiert.

Anmerkung

- Zur Implementierung eines ADT werden typischerweise Datenstrukturen aus vorhandenen Datentypen gebildet, z.B. int A [] = new int[10]; int length = 0;

Beispiele

List Abstract Data Type

Contains elements of same type arranged in sequential order

Initialize( ) - Initialize the list to be empty.
get( ) - Return an element from the list at any given position.
insert( ) - Insert a new element at any position of the list.
remove( ) - Remove first occurrence of any element from a non empty list.
removeAt( ) - Remove the element at a specified location from a non empty list.
replace( ) - Replace an element at any position by another element.
size( ) - Return the number of elements in the list.
isEmpty( ) - Return true if the list is empty, otherwise return false.
isFull( ) - Return true if the list is full, otherwise return false.

Stack Abstract Data Type

Contains elements of same type arranged in sequential order

Initailize( ) - Initialize the stack to be empty.
Push( ) - Insert an element at one end of the stack called top.
Pop( ) - Remove and return the element at the top of the stack
Peek( ) - Return the element at the top of the stack without removing it
size( ) - Return the number of elements in the stack.
isEmpty( ) - Return true if stack is empty
isFull( ) - Return true if no more elements can be pushed

C# unterstützt den Umgang mit ADT's durch die Bereitstellung von Klassen und Interfaces.

List und Stack kann nun konkret als Array oder als verkettete Liste implementiert werden

Die verwendete Implementierung hängt von der konkreten Anwendung ab:

- Datenvolumen
- Geschwindigkeit

Abstrakte Datentypen vs. Datenstrukturen

- Datenstruktur ist die physikalische Implementierung eines abstrakten Datentyp

Abstrakter Datentyp
Logische Sicht auf die Daten und den Operationen um die Daten zu manipulieren

Datenstruktur
Konkrete Repräsentation der Daten und Algorithmen um die Daten zu manipulieren

Gruppen von Datenstrukturen

- Linear
  - List
  - Stack
  - Queue
- Tree
  - Binary Tree
  - Balanced Tree
- Dictionary
  - Key-Value Pair
  - Hashtable

Implementierte Datenstrukturen in C#

- Array
- ArrayList
- List
- LinkedList
- Dictionary
- Hashtable
- HashSet
- Stack
- Queue

Aufgabe

- Recherchieren Sie auf https://source.dot.net/ folgende Fragen:
  - Auf welcher Basis ist die Klasse Queue implementiert? D.h. mit welchem Datentyp speichert die Klasse die Daten? Array
  - Auf welchem Basis-Datentyp basiert die Klasse ArrayList? Array
    - Was passiert, wenn die Kapazität erreicht ist? Verdoppelt seine grösse