データ構造とアルゴリズム
（第2回）

モバイルコンピューティング研究室
柴田史久



1

1

本日の講義内容

- 基本的なデータ構造（１）
  - リスト
  - スタック
  - 待ち行列（キュー）（今日は言葉だけ）
- 配列によるデータ構造の実現

2

2

教科書 第4章（pp.77～84）

基本的なデータ構造(1)

3

3

リスト

- 最も基本的なデータ構造
  - 一覧表，目録，名簿
- 要素を順番に並べたもの
  - 並び，列（sequence）とも
- 連結リスト（linked list）ではない

4

4

抽象データ型としてのリスト

- N個の要素を順番（線形）に並べたもの
- 要素は順序付けされている
- 線形リスト（linear list）とも
- k 番目の要素を $x[k]$ と書くならば，
  - 前の要素は $x[k-1]$，次の要素は $x[k+1]$
  - 先頭の要素は $x[1]$，最後の要素は $x[n]$

5

5

リストに対する基本操作

項番	操作
1	k番目の要素の前に要素を挿入
2	k番目の要素を削除
3	k番目の要素の内容を読む／書く
4	特定のキーをもつ要素を探索
5	複数のリストを1つにまとめる
6	1つのリストを複数のリストに分割
7	リストの複製
8	リストに含まれる要素の個数を求める

- すべてが必要になるわけではない
- すべて効率よく実行できるデータ構造は困難

6

6

特別なリスト

- 先頭と末尾に挿入と削除が行われるリスト
  - スタック（stack）
  - 待ち行列（キュー）（queue）

7

7

スタック(stack)

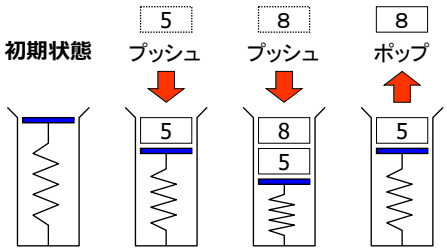
- リストの先頭のみで挿入と削除ができる
  - 一番最後に挿入された要素が削除の対象
- 別名
  - 棚（あまり聞かない）
  - LIFO（last-in first-out）
  - プッシュダウンリスト（push-down list）

8

8

スタックの概念(1)

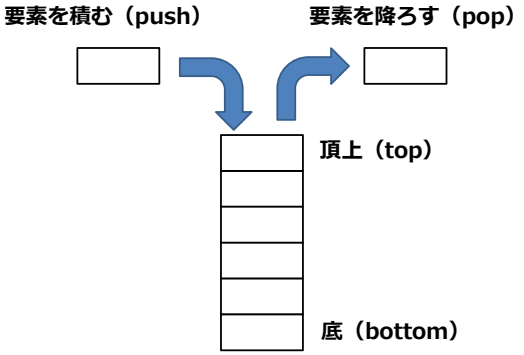
- 挿入と削除をリストの先頭でのみ行う
- 挿入の操作：「プッシュ（積む）」
- 削除の操作：「ポップ（おろす）」



9

9

スタックの概念(2)



10

10

配列によるデータ構造の実現

11

11

配列によるリストの実現

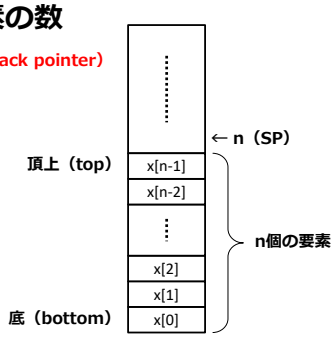
- リストは要素を線形に並べたもの
- 配列を使えばいいのでは？
  - int[] x = new int[100] ;
- 計算量
  - k番目の要素の読み書き: $O(1)$
  - k番目への要素の追加： $O(n)$
  - k番目の要素の削除： $O(n)$
- 一般的にはリストの実現に配列は不適
  - 特殊な例（スタック，待ち行列）には適用可能

12

12

配列によるスタックの実現

- 一般的なリストとは逆向きに要素を入れる
- n はスタックの要素の数
  - スタックポインタ（stack pointer）
- プッシュ（積む）
  - x[n++] = data ;
- ポップ（降ろす）
  - data = x[--n] ;



13

13

配列によるスタックの計算量

- 要素の挿入（プッシュ）：$O(1)$
  - x[n++] = data ;
- 要素の削除（ポップ）：$O(1)$
  - data = x[--n] ;
- スタック容量の検査
  - いっぱいかどうか：$O(1)$
  - 空かどうか：$O(1)$

14

14

スタックを実現するクラス(1)

- double型の値を入れるDoubleStackクラス
- 準備
  - コンストラクタでスタックの最大サイズを指定
  - double型の配列にデータを格納
  - 外部からデータにアクセスできないように設定

```
public class DoubleStack {
  private double[] dataArray ;  // データを格納する配列
  private int sp ; // スタックポインタ = 格納されているデータの個数

  public DoubleStack(int maxSize) {  // コンストラクタ
    // ここでデータを格納する配列の領域を確保し，初期化
    // スタックポインタを初期化
  }
}
```

15

15

補足:修飾子について

- Javaではクラスや変数，メソッド等に修飾子をつけてアクセスを制御できる
  - 意図しない変数値の変更などを防ぐため
  - カプセル化に重要な機能
- 例：変数の修飾子

キーワード	内容
private	同じクラスからしかアクセスできない変数
無指定	同じパッケージからしかアクセスできない変数
protected	同じパッケージ，またはサブクラスからしかアクセスできない変数
public	どこからでもアクセスできる変数

16

16

スタックを実現するクラス(2)

- スタックに対する操作
  - 容量の検査
    - isFullメソッド：スタックがいっぱいかどうか
    - isEmptyメソッド：スタックが空かどうか
    - sizeメソッド：スタックに格納されたデータ数
  - 要素の挿入・削除
    - pushメソッド：スタックに要素を積む（挿入）
    - popメソッド：スタックから要素を降ろす（削除）
    - clearメソッド：スタックの内容を消去
  - 内容の確認
    - showメソッド：スタック内のデータを表示

17

17

スタックを実現するクラス(3)

- isFullメソッド
  - 格納されているデータの個数と配列サイズを比較
  - 配列のサイズは dataArray.length
- isEmptyメソッド
  - スタックポインタの値で判断
- sizeメソッド
  - データの個数＝スタックポインタの値

18

18

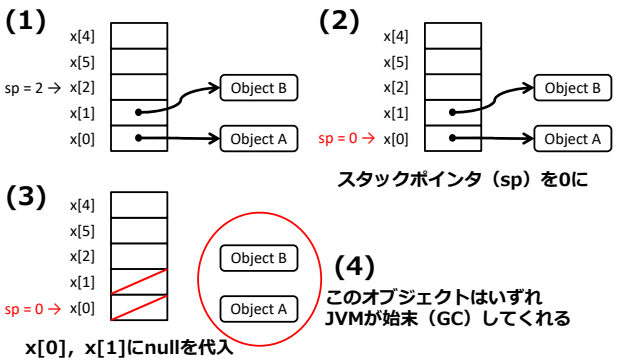
スタックを実現するクラス(4)

- pushメソッド
  - スタックがいっぱいでないかを確認
  - いっぱいでなければ末尾にデータを追加
- popメソッド
  - スタックが空でないかを確認
  - 空でなければ末尾からデータを取り出し
  - 空いた要素を初期化

19

19

スタックを空にする際の問題点



20

20

スタックを利用した計算

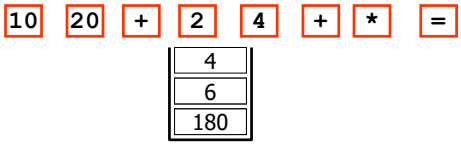
- 数式の計算
  - (10 + 20) × (2 + 4)
- これを計算すると
  - 30 × 6 = 180
- 「10と20を足したものに，2と4を足したものをかける」
- 逆ポーランド記法（Reverse Polish Notation; RPN）
  - 10 20 + 2 4 + * =

21

21

スタックを利用した計算

1. 数値が現れたら，その数値をスタックに積む
2. 演算子（+，－，*，/）が現れたら，スタックから２つの数値を取り出し，演算子に対応した計算を行い，その結果をスタックに積む
3. イコール（=）が現れたら，スタックから１つの数値を取り出し，それを結果として表示



22

22

まとめ

- 基本的なデータ構造（１）
  - リスト
  - スタック
- 配列によるデータ構造の実現
  - 配列によるスタック
  - スタックを利用した計算
    - 逆ポーランド記法

23

23

参考文献

- 定本 Javaプログラマのための アルゴリズムとデータ構造（近藤嘉雪）
- 新・明解 Javaで学ぶ アルゴリズムとデータ構造（柴田望洋）
- 岩波講座ソフトウェア科学 3 アルゴリズムとデータ構造（石畑清）
- Javaで学ぶアルゴリズムとデータ構造 Robert Lafore（著）・岩谷 宏（翻訳）
- Java アルゴリズム+データ構造完全制覇 オングス (著)・杉山 貴章・後藤 大地 (監修)

24