基本情報技術Ⅰ

8. データ構造

菊池浩明

講義目標

- 教科書
  - 2-1 データ構造
    - » 1. 配列
    - » 2. リスト構造
    - » 3. スタックとキュー

1. 配列

- データ構造
  - データの格納方法.
    - » 配列，リスト，木，ヒープ
  - プログラミング＝アルゴリズム＋データ構造
  - 基本操作
    - » 1. 参照
    - » 2. 挿入
    - » 3. 削除
    - » (4. 更新，5. 探索）

配列の構造

- 1次元配列

A

A[0]	A[1]	A[2]
15	10	40

- int A[ ] = {15,10,40}

- 2次元配列

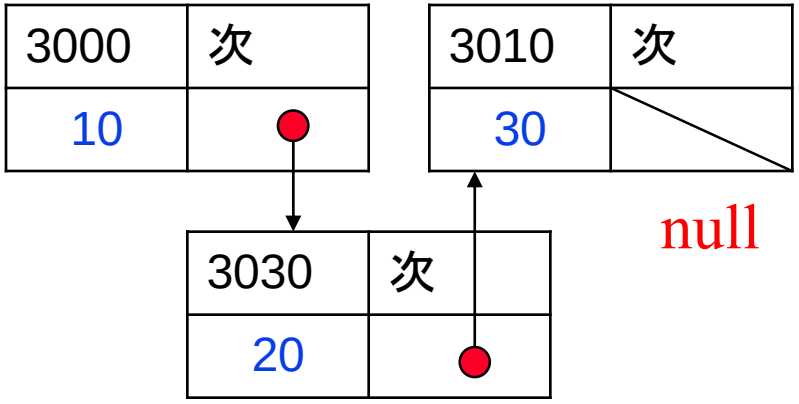
B	B[0][0]	B[0][1]	B[0][2]
B[0]	15	10	40
B[1]	25	20	50

- int A[ ][ ] =
  {{15, 10, 40},
  {25, 20, 50}}

2. リスト★★頻出

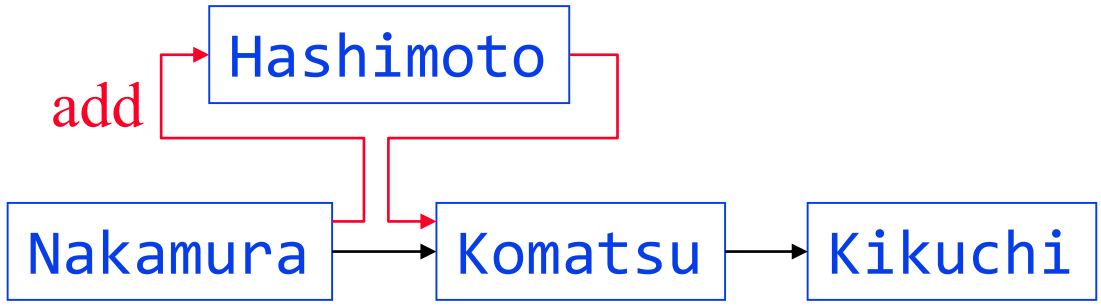
- 配列
  - データが順番に並んでいるデータ構造
  - 静的（配列長を後から変更できない）
- (連結)リスト構造
  - 次のデータを示すポインタで構成される.
  - 動的（配列長を変更可能）

2000	2001	2002
A[0]	A[1]	A[2]



List クラス仕様 (processing)

- コンストラクター
  - ArrayList<型> = new ArrayList<型>();
  - 型： ジェネリック (generic)　汎用の
- メソッド
  - void add(型 要素);　　要素の挿入
  - 型 get(int index);　　index番目要素の参照
  - 型 remove(int index)　要素の削除



もしも配列で作ると

- ListNames.pde Array版

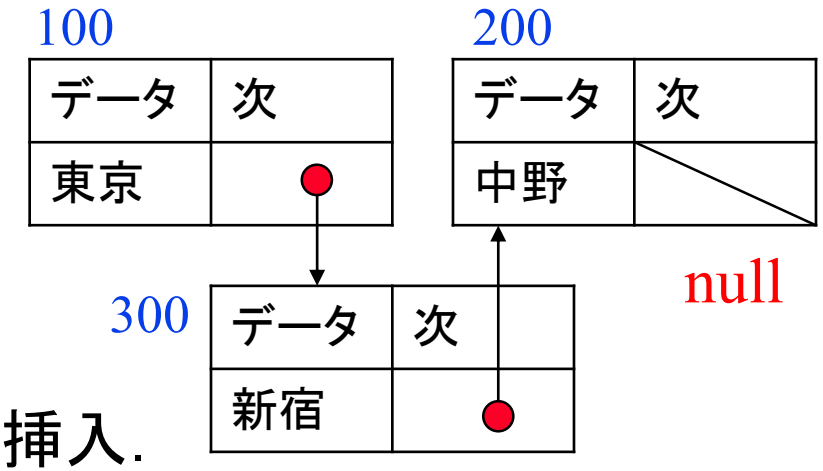
```
1.  String s[] = new String[10];
2.
3.          s[0] = “Nakamura";
4.          s[1] = “Komatsu";
5.          s[2] = "Kikuchi";
6.          for(int i = 0; i < s.length; ++i){
7.                  println(s[i]);
8.          }
9.
10.         s[3] = s[2];
11.         s[2] = s[1];
12.         s[1] = “Hashimoto";
13.         for(String a : s){
14.                 println(a);
15.         }
```

リストの長さに比例する処理（効率悪）

リストの実現方法

- 配列（メモリ）

アドレス	データ	次
100	東京	300
200	中野	0 (null)
300	新宿	200



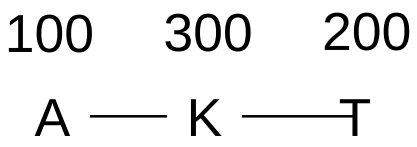
- 新宿の前に御茶ノ水を挿入.

アドレス	データ	次
100	東京	300
200	中野	0
300	新宿	200
400	御茶ノ水	

例1

- 双方向のポインタを持つリスト構造がある．新たな社員GをKの後に追加せよ．

アドレス	社員	次ポインタ	前ポインタ
100	A	300	0
200	T	0	300
300	K	200	100
400	G		



例2.

- 数列 2, 3, 5, 7, 11 がリストに格納されている. ポインタを参照する回数を求めよ.
  - 3を参照する.　　2回
  - 11を参照する
  - 先頭に1を追加　　1回
  - 先頭要素を削除
  - 末尾に13を追加

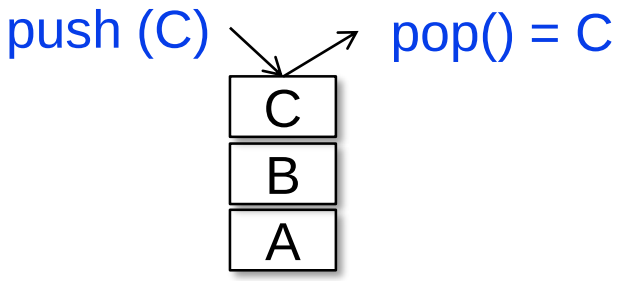
性能の整理

- 長さ n = 10の数列のx = 6番目のデータを対象とする時の処理時間（アクセスの回数）をもとめよ.

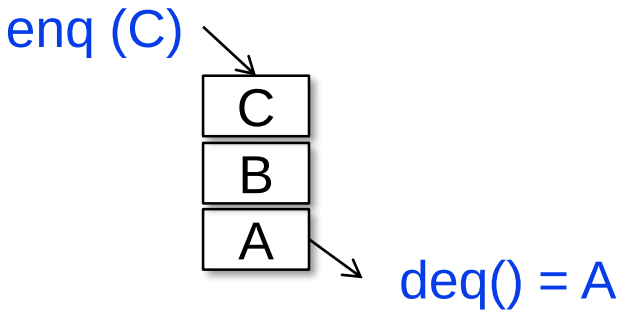
	配列	リスト
参照	1回	6 回 (x回)
削除	4回 (n-x回コピー)	1回 (+x回参照)
更新		
挿入		

3. スタックとキュー ★★頻出

- Stack
  - push(x) xを格納
  - pop()　データ取出し
  - 後入れ先出し
    Last-In First-Out
    (LIFO)

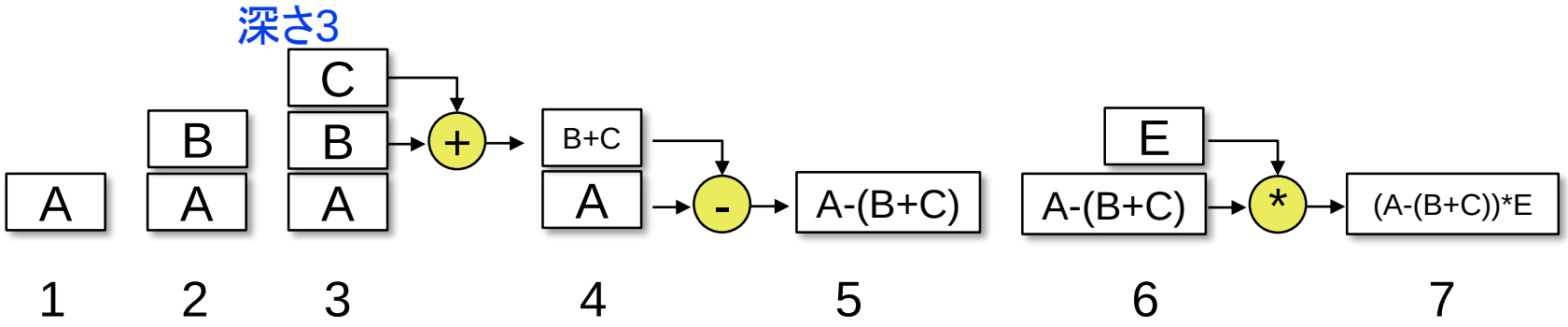


- Queue
  - enq(x) xを格納
  - deq()　データ取出し
  - 先入れ先出し
    First-in First-Out
    (FIFO)



(復習) 逆ポーランド表記法

- 後置表記法
  - 演算式を表す表記法の一つ．スタック（先入れ後出しデータ構造）を用いて，括弧を省略する．
  - 1. オペランド(変数や値）はスタックにpush（積む）
  - 2. 演算子(+, *, =)はスタックから要素を二つpop(下す)して，演算を行い，結果をpushする．
  - 例) ABC+-E* は E*(A-(C+B)) と同値



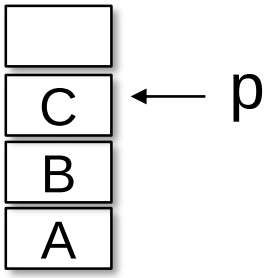
キューの例

- 処理時間の差のある入出力装置間の調整（バッファ）
  - 例）プリントキュー（ファイルをenq, プリンタがdeq)
  - 例）メールキュー（メールをenq, メールサーバが定期的にdeq)
  - 例）ネットワーク通信（データをenq, ソケットがネットワークに送信）

スタックとキューの実現

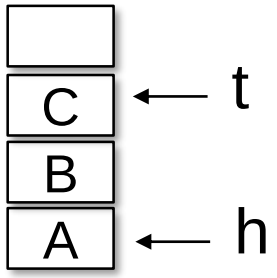
- スタック

```
1.  push(x){
2.      ++p;
3.      A[p] = x;
4.  }
5.  pop(){
6.     x = A[p];
7.     --p;
8.     return x;
9.  }
```



- キュー

```
1.  enq(x){
2.      ++t;
3.      A[t] = x;
4.  }
5.  deq(){
6.     x = A[h];
7.     --h;
8.     return x;
9.  }
```



例3

- enq(1), enq(2), deq(), enq(3), deq(), deq() した時の値を求めよ.

- push(a), push(b), pop(), push(c), pop() した時の値を求めよ.

例4.

- A, B, C, Dの順に到着するデータに対して，一つのスタックだけを用いて出力可能かどうか示せ.
  - ア A, D, B, C
  - イ B, D, A, C
  - ウ D, C, A, B
  - エ C, B, D, A

ヒント

- A, Bの入力
  - push(A), pop, push(B), pop → A, B
  - push(A), push(B), pop, pop → B, A
- A, B, Cの入力
  - push(A),pop, push(B), pop, push(C), pop→A,B,C
  - push(A),pop, push(B), push(C), pop, pop→A,C,B
  - push(A),push(B), push(C), pop, pop, pop→C,B,A
  - push(A),push(B), pop, push(C), pop, pop→B,C,A
  - push(A),push(B), pop, pop, push(C), pop→B,A,C

宿題

- 練習問題 p.102
  - 2.

まとめ

- プログラミングはアルゴリズム＋（　　）.
- リスト構造は，格納場所（　　）とデータと次の要素を示す（　　）で構成される．参照は長さに比例するが，配列に比べて（　）と削除が早い（1回のアクセス）.
- スタックは，後入れ先出し (　　)型のデータ構造で，pushで格納，（　　）で取り出す.
- （　　）は，先入れ先出しFIFO型のデータ構造であり，(　　)で格納，deqで取り出す.