アルゴリズムとデータ構造

第2回　基本的な抽象データ型

http://www-ui.is.s.u-tokyo.ac.jp/~takeo/course/

五十嵐 健夫

takeo@acm.org

前回の内容

イントロダクション
モデル化の例（交差点）
抽象データ型
計算量（Oと Ω）

今回の内容

リスト
スタック
待ち行列
写像、連想記憶
再帰呼び出し

抽象データ型としての「列」

$a_1, a_2, a_3, \ldots, a_{1n}$　線形順序

insert, indexOf, get, remove, next, prev, clear, first, print, end

Purge の例（重複を取り除く）

列の実現

配列による実現

○ ランダムアクセス　× 挿入と削除

ポインタによる実現（通常のリスト）

× ランダムアクセス　○ 挿入と削除

配列

○ ランダムアクセス　× 挿入と削除

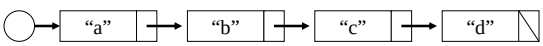
String[] labels = {“a”,”b”,”c” ,”d”}

“a”
“b”
“c”
”d”

リスト

× ランダムアクセス　○ 挿入と削除

```
LinkedList labels = new LinkedList();
labels.add(“a”);
labels.add(“b”);
labels.add(“c”);
labels.add(“d”);
```



スタック

clear, pop, push, empty

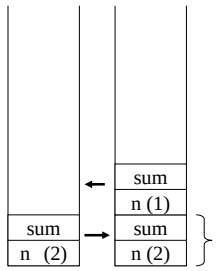
例（消去文字の処理、再帰呼び出し）

実装　（ポインタと配列）

スタックと再帰呼び出し

$$\sum_{i=1}^{n} i^2$$

```
int foo(int n){
   if (n == 1) return 1
   int sum = foo(n-1)+n*n;
   return sum;
}
```



活動レコード

待ち行列 (Queue)

clear, front, enqueue,
dequeue, empty

例（イベントキュー、データ転送

実装　（ポインタと循環配列）

連想記憶　(Map)

clear, put, get

例（社員）

実装　（配列とポインタ）

再帰呼び出しの繰り返しへの変換

コード中に自分でスタックを定義する。

呼び出すときは
- **スタックへ変数保存**
- **スタックへ戻り先保存**
- **呼び出し先の変数の設定**
- **先頭へJUMP**

戻るときは
- **スタックから状態復元**
- **スタックから戻り先復元**
- **戻り先へJUMP**

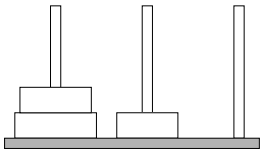
（早くなるが読みにくくなる）

```
int foo(int n){
   if (n == 1) return 1
   int sum = foo(n-1)+n*n;
   return sum;
}
```

```
int foo(int n){
        push(end_point)
        push(n)
        entry_point
                if (n == 1) {
                        returned_value = 1;
                        goto return_point
                }
                push(resume_point)
                push(n);
                n = n-1;
                goto entry_point;
        resume_point:
                int sum = returned_value + n*n;
                return_value = sum;
                goto return_point;
        return_point:
                n = pop();
                return_position = pop();
                goto return_position;
        end_point:
                return returned_value;
}
```

ハノイの塔

ピンsource**からn枚円盤を**
ピンtarget**へ移す。**



```
void hanoi(int n, int source, int target){
   if (n == 1) move(source, target);
   else {
        int middle = 6- source - target; //空ピン
        hanoi(n-1, source, middle);
        move(source,target);
        hanoi(n-1, middle, target);
   }
}
```

計算量 $O(2^n)$

```
hanoi(int n, int source, int target){
        push(n, source, target, middle, end_point)
        entry_point
                if (n == 1) {
                        move(source, target);
                        goto return_point
                }
                int middle = 6-source-target //空ピン
                push(n, source, target, middle, resume_point_1);
                n = n-1; source = source; target = middle;
                goto entry_point;
        resume_point_1:
                move(source, target);
                push(n, source, target, middle, resume_point_2);
                n = n-1; source = middle; target = target;
                goto entry_point;
        resume_point_2:
                goto_return_point;
        return_point:
                n, source, target, middle, return_position = pop();
                goto return_position;
        end_point:
                return;
}
```

まとめ

配列
リスト
スタック
待ち行列
連想記憶

（再帰呼び出しの除去）