アルゴリズム論

2017年6月26日

樋口文人

目次

- ポインタ
  - 使い方
    - 変数とアドレス
    - 関数

ポインタ変数

ポインタ

- 変数のアドレスを表す変数
  - 通常の変数

    int x = 0; // x という名前の変数に値 0 を設定
    int *a; // a は整数型の変数へのポインタ
    a = &x; // &x は変数 x のアドレスを表す
  - 例

```
int x, y, z = 1;
int *a, *b, *c;
a = &x; b = &y; c = &z;
*a = *c + *c;
*b = *a + *c;
printf("%i, %i, %i", x, y, z);
```

malloc()

- プログラムからメモリ上の領域を確保してその先頭アドレスを返す関数
- 使い方（int型の領域を確保する場合）
  - #include <stdlib.h> が必要
  - (int *)malloc(sizeof(int))

```
int *new;
new = (int *)malloc(sizeof(int));
```

- 不要になったメモリ領域はfree()関数で解放する
  - free(解放するメモリ領域へのポインタ)

```
free(new);
```

メモリ内の変数

a = &x;

c = &z;

&x

*a

a

変数名	アドレス	内容
	101	
int x	102	0
int y	103	
int z	104	1
	105	
	106	
int *a	107	102
int *b	108	NULL
int *c	109	104
	110	
	111	

メモリ内の変数

a = &x;

c = &z;

&z

*c

c

変数名	アドレス	内容
	101	
int x	102	0
int y	103	
int z	104	1
	105	
	106	
int *a	107	102
int *b	108	NULL
int *c	109	104
	110	
	111	

問題のある参照

変数名	アドレス	内容
	101	
int x	102	0
int y	103	
int z	104	1
	105	
	106	
int *a	107	102
int *b	108	NULL
int *c	109	104
	110	
	111	

?
*b

b

関数の副作用

- 関数 y = f(x)
  - f(入力) = 出力
  - x: 入力 → 引数
  - y: 出力 → 返り値
- 副作用
  - 入力に対して，出力を返す以外に起こる変化

```
int i = 0;
int f(int x){
    i++;
    return x*x;
}
```

関数の内外での情報のやり取り

- 基本：引数を与えて出力を得る
  - 引数は値渡し
  - 仮引数を含め，内部の変数はローカル変数
  - 関数の実行終了後は消滅
- グローバル変数
  - 複数の関数での情報の共有
  - 前回の実行結果を保持した関数の実行
    - 前回の続きからカウントする，など
  - 複数の関数がグローバル変数の書き換えを行うのは問題が多い

関数の引数とポインタ変数

- 通常の変数

  int i=3;

  f(x){x=5;} f(i)でiは5にならない

  i=f(x){x=5; return x;} iは5になる

- ポインタ

  int *pi=3;

  f(int *px){ *px=5; } f(pi) で*piは5になる

  ポインタ x を通じて i そのものは変えられない

- ポインタのポインタ

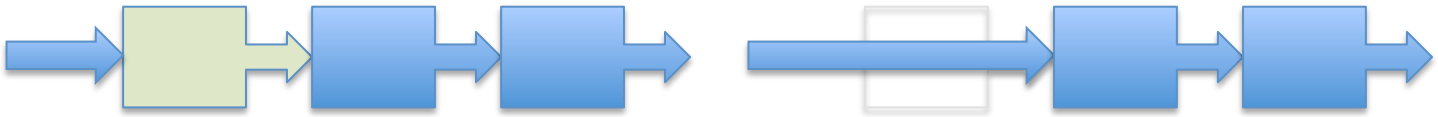
  int **ppi; ppi = π

  f(int **ppx) { **ppx=5; }

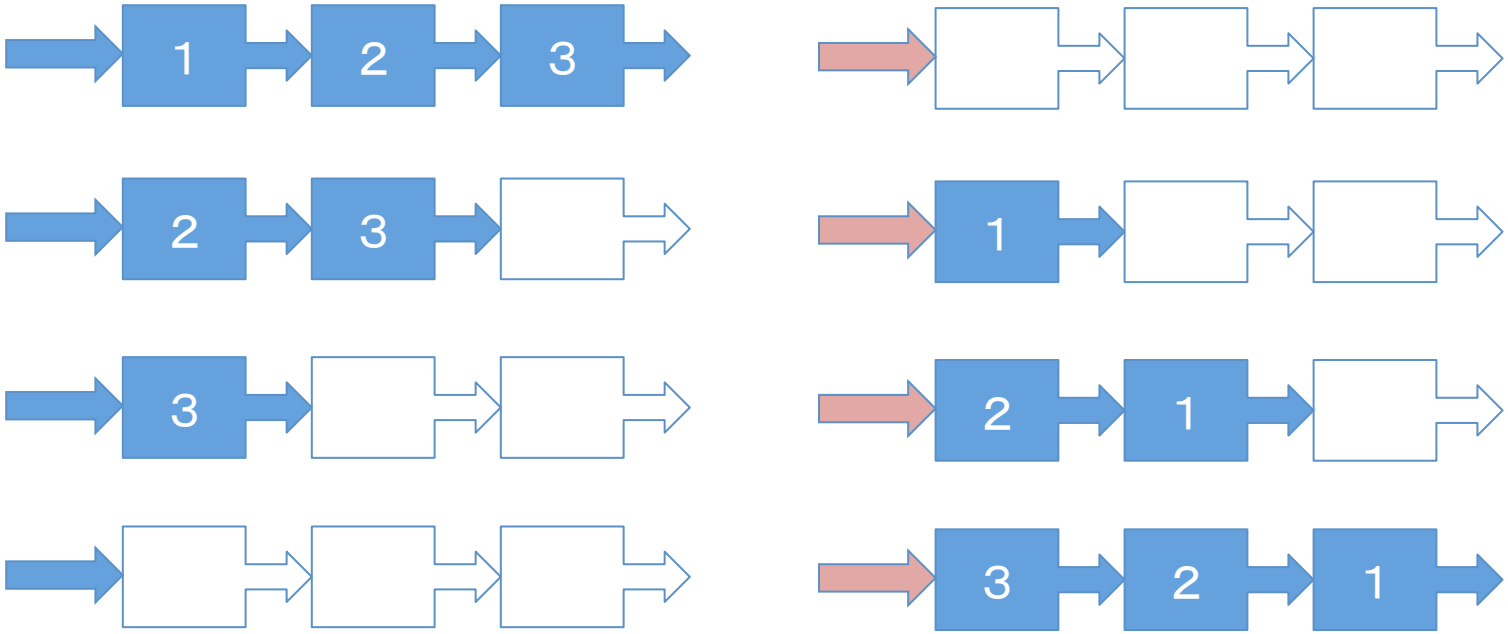
  f(ppi) で**ppiは5になる, *ppi も変えられる

まとめ

- 線形リストの先頭への挿入や銭湯からの削除では線形リストを指し示すポインタ変数の内容を書き換える必要がある
  - 引数に線形リストへのポインタを使う場合
    - 関数の副作用では書き換え不可
    - 返り値を使う
  - 引数にポインタのポインタを使う場合
    - 関数の副作用で書き換え可能
    - 返り値を使わなくともよい



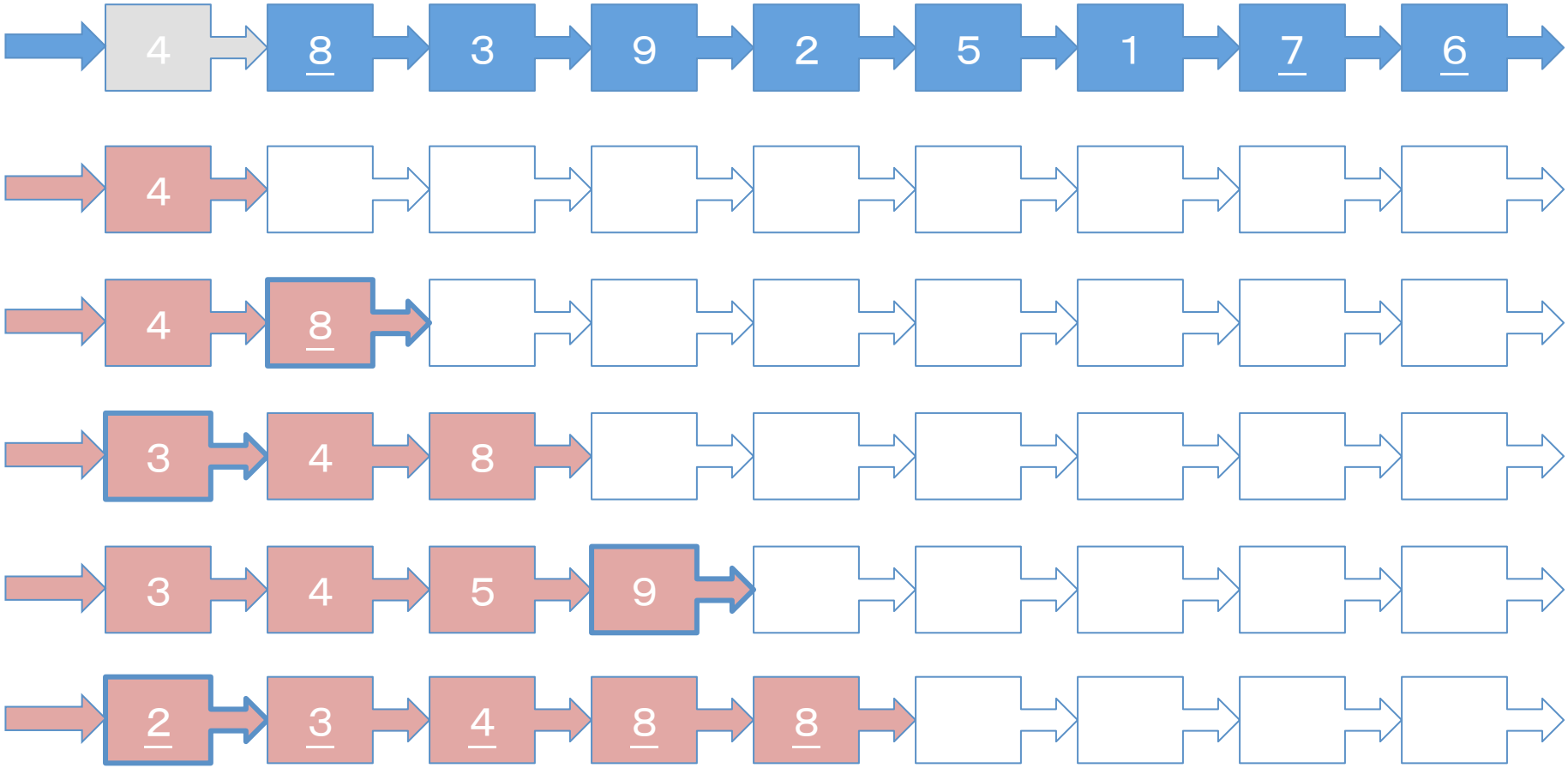
線形リストの繋ぎ変え



線形リストの先頭（末尾）から取って，先頭（末尾）に追加すると逆順

スタック：線形リストの先頭から追加し，先頭から取り去る
キュー：線形リストの先頭から追加し，末尾から取り去る

挿入ソート



バックトラック

- 深さ優先探索の一種
- 選択肢を選ぶ
- 選択した状況で選択肢を選ぶ
  - これを繰り返す
  - 一つの状況で選択肢がなくなったら一つ前の状況に戻る

やってみよう：作業

- 線形リストでの整列アルゴリズムを考える
- フローチャートと描いてみる
- 引数に何を使うか？
  - ポインタ
  - ポインタのポインタ
  - 返り値

ポインタの使い方：サンプル1

```
// pointer_eg1.c
//
// ポインタ変数　サンプルプログラム1
// 〜「変数のアドレス」と「ポインタが指すアドレスの中身」
//
// Created by Fumito Higuchi
// Date: 2017-06-12.

#include <stdio.h>

int main(void){
  int x;
  int y = 1;
  int z = 2;

  int *a, *b, *c;  // ポインタ変数の宣言

  a = &x; // ポインタ a, b, c の各値は変数 x, y, z のアドレス
  b = &y;
  c = &z;

  // 6つの変数の値を比較する
  printf("通常の変数: %i, %i, %i\n",  x,  y,  z);
  printf(" ポインタ: %i, %i, %i\n", *a, *b, *c);
  *a = *b + *c; // ポインタが指すアドレスの中身を演算
  printf("通常の変数: %i, %i, %i\n",  x,  y,  z);
  printf(" ポインタ: %i, %i, %i\n", *a, *b, *c);
}
```

ポインタの使い方：サンプル2

```
// pointer_eg2.c
//
// ポインタ変数　サンプルプログラム2
// ～ポインタを含む構造体による線形リスト　個別に作って繋ぐ
//
// Created by Fumito Higuchi
// Date: 2017-06-12.

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void){

struct cell {
struct cell *next; // 次のセルを指し示すためのポインタ
int val;
};

struct cell A, B, C, D, E, F, G, H, I, J; // 10個のセルを宣言

// 個々のセルを相互に繋ぐ
A.next = &B; // セルAからセルBを後続として指し示す
A.val  = 9;
B.next = &C; // セルBからセルCを後続として指し示す
B.val  = 8;
C.next = &D;
C.val  = 7;
D.next = &E;
D.val  = 6;
E.next = &F;
E.val  = 5;
F.next = &G;
F.val  = 4;
G.next = &H;
G.val  = 3;
H.next = &I;
H.val  = 2;
I.next = &J;
I.val  = 1;
J.next = NULL; // セルJに後続は無いので、NULLを指定
J.val  = 0;

struct cell *p; // ポインタ p は線形リストの先頭(セルA)を指し示す
p = &A;

// 線形リストの先頭から順にセルを辿って内容(val)を出力する
while (p != NULL) {
printf("%i", p->val);
p = p -> next; // p が次のセルを指し示すように更新
}
printf("\n線形リストの内容を出力しました。\n");
}
```

ポインタの使い方：サンプル3

```
// pointer_eg3.c
//
// ポインタ変数　サンプルプログラム3
// ～「線形リストの連続生成」と「乱数と malloc()の使用」
//
// Created by Fumito Higuchi
// Date: 2017-06-12.

#include <stdio.h>
#include <stdlib.h> // malloc(), rand()を利用するのに必要
#include <time.h>   // srand()の引数にシステムクロックを利用するのに必要

int main(void){

srand((unsigned)time(NULL)); // 乱数の初期化

struct cell {
struct cell *next;
int val;
};

struct cell *p, *q, top;
q = &top;
top.next = NULL;       // 最初のセルを設定（後続のセルは無い）
top.val  = rand()%32; // 乱数で val の値を設定
                       // 値が大きくなり過ぎないように剰余を使っている
int i = 1;             // 作成するセルの個数を数えるためのカウンタ変数

while (i < 10){
// セルの自動生成（メモリから必要なサイズを割り当て、そのアドレスをポインタにセット
// p 新しいセルについてのポインタ
// q 直前に作ったセルについてのポインタ
p = (struct cell *)malloc(sizeof(struct cell));
p->next = NULL;
p->val  = rand()%32;
q->next = p; // 直前のセルの後ろに新しいセルを繋ぐ
q = p;
i++;
}

p = &top;
while (p != NULL) {
printf("%i\n", p->val);
p = p -> next;
}
printf("\n線形リストの出力終了\n生成できる乱数の最大値： %d\n", RAND_MAX);
}
```

宿題：ex10

- C言語で実装したポインターを使った線形リストで整列プログラムを実装してください
  - リストに保存されるデータ数は10個程度で良い
  - 乱数で生成したデータを持つ線形リストを作る
  - 整列前のデータを表示
  - 整列
    - 整列アルゴリズムは任意
  - 整列後のデータを表示

提出についての注意

- プログラム
  - 提出はプログラムのソースファイル(.c)のみ
  - higuchi_fumito_ex10 のように氏名と宿題番号にすること
  - higuchi_fumito_c5_ex01 (同姓同名はクラスを付加）
- プログラムの冒頭に氏名、学年、クラス、番号等をコメントとして記入
  - 日本語の文字コードはutf-8が望ましい
  - 参考にした資料の他、簡単な感想も付け加えてください
- Oh-o!Meijiから提出（次回の授業開始までに）

連絡先

樋口文人

wenren@meiji.ac.jp