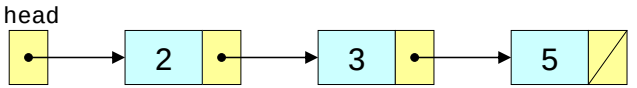
プログラミング演習
構造体(3)

阿萬裕久
aman@cs.ehime-u.ac.jp

例題1

- 下図のようにデータが数珠つなぎに連なった構造を**リスト構造**という.



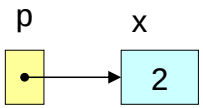
- ここでの矢印による連結をポインタによって実現する. この構造を構造体 node を使って実現し, データの格納と表示を行いなさい.

ポインタによるリンク

- まず, int 型データをポインタによってアクセスする場合:

```
int  x;
int* p;

p = &x;
*p = 2;
```



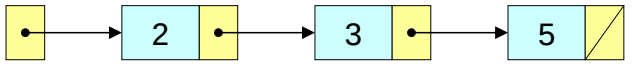
参照先(x)が int 型なので, ポインタ変数 p は

`int*`

として宣言

ポインタによるリンク

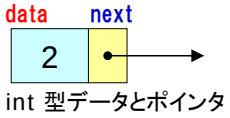
- ところが, この例題で作りたいのは, 単発ではなく連続したリンク



- つまり, 参照先は単純な int 型ではなく, その後ろへのリンク(ポインタ)も必要

作りたいのは新しい型（構造体）

- では，実際に作りたい型を構造体で考える
- 構造体の中身は
  - データ部分の data
  - ポインタ部分の next



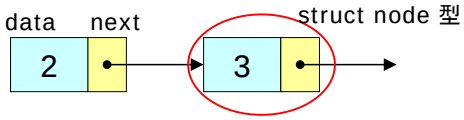
int 型データとポインタ

```
struct node {
  int  data;
       * next;
};
```

参照先の
データ型
を書けばよい

参照先もまた「struct node」

- next の参照先には，同じ struct node 型のデータがくるので




```
struct node {
  int  data;
  struct node * next;
};
```

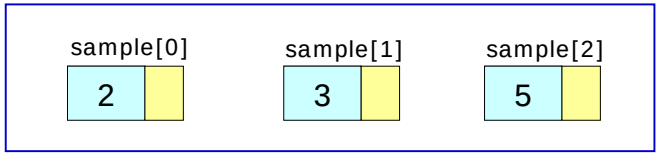

例題1：解説

Web ページのコーディング例を参照

- まず，3つの node を作成

15行目
```
struct  node  sample[3];
```

20～22行目
```
sample[0].data = 2;
sample[1].data = 3;
sample[2].data = 5;
```



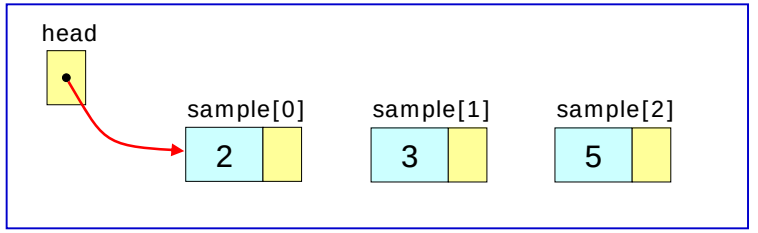
例題1：解説

Web ページのコーディング例を参照

- 次に，先頭 node をリンクでつなぐ

25行目
```
head = &sample[0];
```

sample[0] の
番地が head へ
代入される



例題1：解説

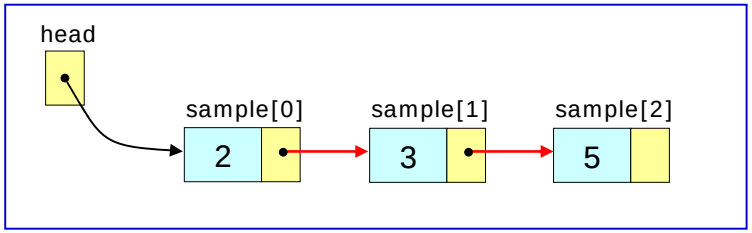
Web ページのコーディング例を参照

- 各 node をリンクでつないでいく

26, 27行目

```
sample[0].next = &sample[1];
sample[1].next = &sample[2];
```

sample[1] の番地が sample[0] のnext 部へ代入される



例題1：解説

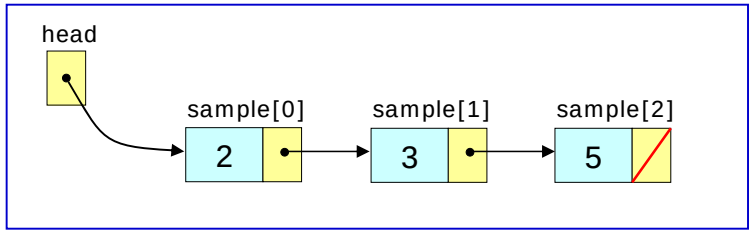
Web ページのコーディング例を参照

- 3つ目の node を終端にする

28行目

```
sample[2].next = NULL;
```

sample[2] のnext 部がどこも指していないことを明言する



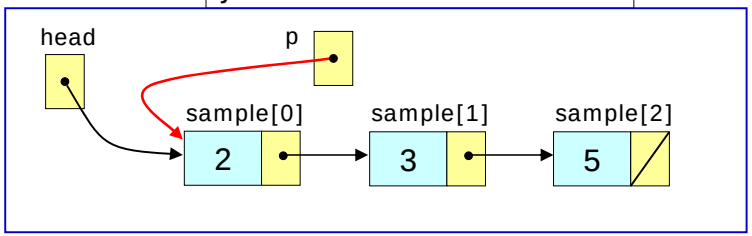
例題1：解説

Web ページのコーディング例を参照

- 先頭から順に表示していく

31～35行目

```
p = head;
while ( p != NULL ){
   printf("%d ", p->data);
   p = p->next;
}
```



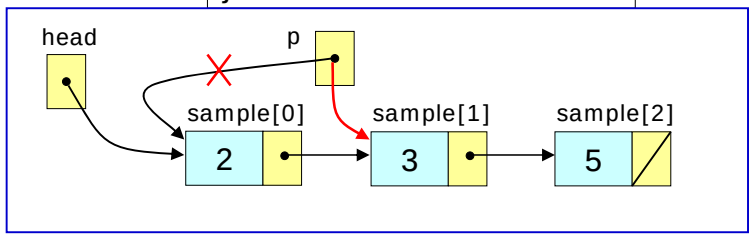
例題1：解説

Web ページのコーディング例を参照

- 先頭から順に表示していく

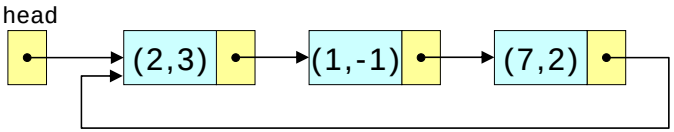
31～35行目

```
p = head;
while ( p != NULL ){
   printf("%d ", p->data);
   p = p->next;
}
```



例題2

- 次のイメージ図に対応したリスト構造をプログラムで実現しなさい.
  なお, 各のノードのデータには 2 次元ベクトル(x, y いずれも整数値) を使用せよ.

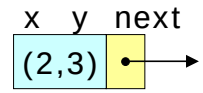


例題2:解説

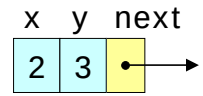
Web ページのコーディング例を参照

- まずは構造体の宣言

```
struct vnode {
  int  x;
  int  y;
  struct vnode* next;
};
```



より正確なイメージだと



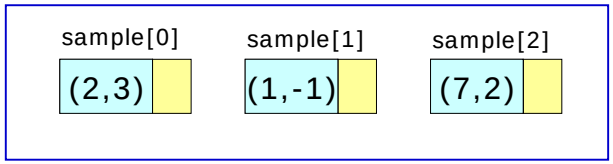
例題2:解説

Web ページのコーディング例を参照

- データの代入

21～23行目

```
sample[0].x = 2;   sample[0].y = 3;
sample[1].x = 1;   sample[1].y = -1;
sample[2].x = 7;   sample[2].y = 2;
```



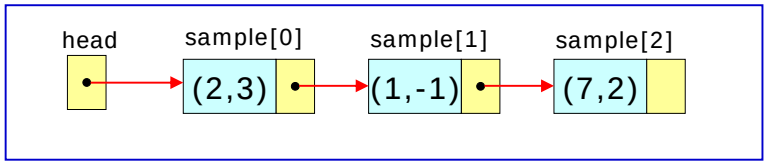
例題2:解説

Web ページのコーディング例を参照

- リンクの生成(その1)

26～28行目

```
head = &sample[0];
sample[0].next = &sample[1];
sample[1].next = &sample[2];
```



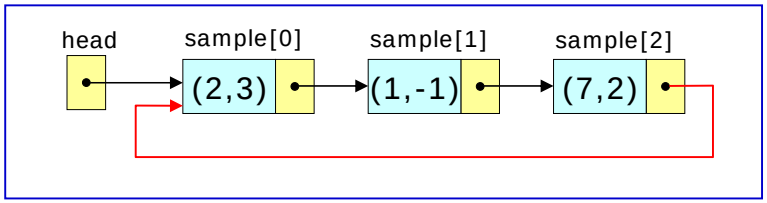
例題2：解説

Web ページのコーディング例を参照

○ リンクの生成（その2）

29行目

```
sample[2].next = head;
```



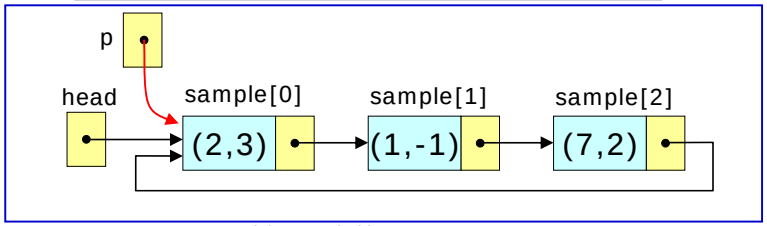
例題2：解説

Web ページのコーディング例を参照

○ データの表示

32～36行目

```
p = head;
while ( p != NULL ){
  printf("(%d,%d)¥n", p->x, p->y);
  p = p->next;
}
```



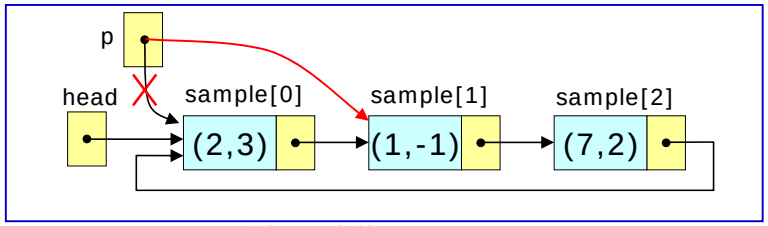
例題2：解説

Web ページのコーディング例を参照

○ データの表示（繰り返しの1回目）

32～36行目

```
p = head;
while ( p != NULL ){
  printf("(%d,%d)¥n", p->x, p->y);
  p = p->next;
}
```



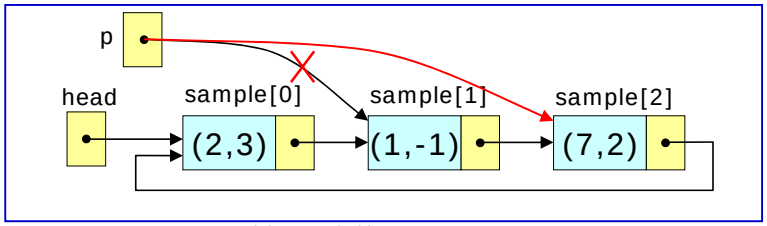
例題2：解説

Web ページのコーディング例を参照

○ データの表示（繰り返しの2回目）

32～36行目

```
p = head;
while ( p != NULL ){
  printf("(%d,%d)¥n", p->x, p->y);
  p = p->next;
}
```



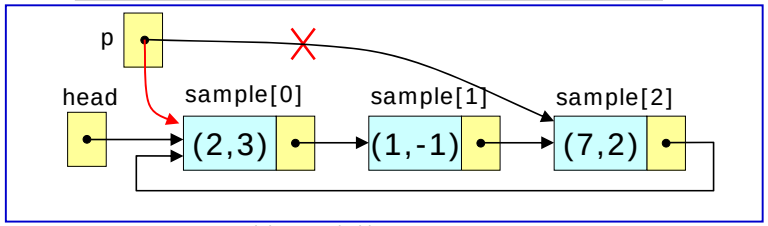
例題2：解説

Web ページのコーディング例を参照

- データの表示（繰り返しの3回目）

32～36行目

```
p = head;
while ( p != NULL ){
   printf("(%d,%d)¥n", p->x, p->y);
   p = p->next;
}
```



例題2：解説

- 循環構造になっているため

  p = p->next

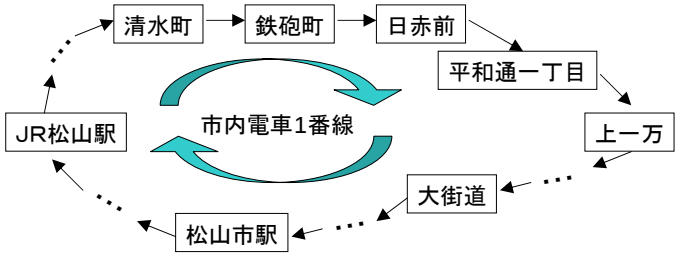
  によって，次のノードへの移動を繰り返すと最初に戻る

条件は常に成立し，無限に繰り返される

```
p = head;
while ( p != NULL ){
   printf("(%d,%d)¥n", p->x, p->y);
   p = p->next;
}
```


例題2：解説

- 循環構造の特徴
  - 環状データやローテーションのモデルとなる
  - 配列と異なり，「その次へ移動」という単調な命令で無限に繰り返すことができる

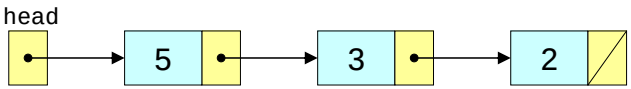


例題3

- 3 つの整数を順に読み込み，それらを逆順に並べたリストを作りなさい．

（例）

入力順：2，3，5



例題3：解説

Web ページのコーディング例を参照

- 基本は例題1と同じ； ポイントは次の部分

21～31行目

```
for ( i = 0; i < 3; i++ ){
   scanf("%d", &x );
   sample[i].data = x;
   if ( i == 0 ){
      sample[i].next = NULL;
   }
   else{
      sample[i].next = &sample[i-1];
   }
}
```


例題3：解説

Web ページのコーディング例を参照

- 1回目（ i == 0 の場合）

21～31行目

```
for ( i = 0; i < 3; i++ ){
   scanf("%d", &x );
   sample[i].data = x;
   if ( i == 0 ){
      sample[i].next = NULL;
   }
   else{
      sample[i].next = &sample[i-1];
   }
}
```

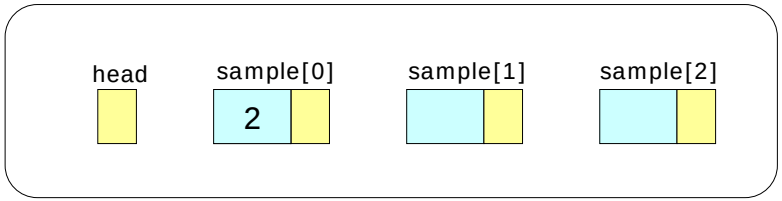
「2」を入力

例題3：解説

Web ページのコーディング例を参照

- 1回目（ i == 0 の場合）

```
scanf("%d", &x );
sample[i].data = x;
```



例題3：解説

Web ページのコーディング例を参照

- 1回目（ i == 0 の場合）

21～31行目

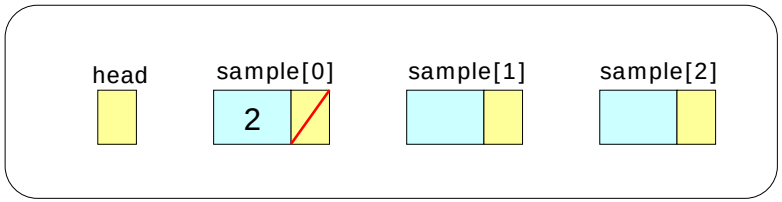
```
for ( i = 0; i < 3; i++ ){
   scanf("%d", &x );
   sample[i].data = x;
   if ( i == 0 ){
      sample[i].next = NULL;
   }
   else{
      sample[i].next = &sample[i-1];
   }
}
```

例題3：解説

Web ページのコーディング例を参照

- 1回目（ i == 0 の場合）

```
if ( i == 0 ){
   sample[i].next = NULL;
}
```



例題3：解説

Web ページのコーディング例を参照

- 2回目（ i == 1 の場合）

21～31行目

```
for ( i = 0; i < 3; i++ ){
   scanf("%d", &x );
   sample[i].data = x;
   if ( i == 0 ){
      sample[i].next = NULL;
   }
   else{
      sample[i].next = &sample[i-1];
   }
}
```

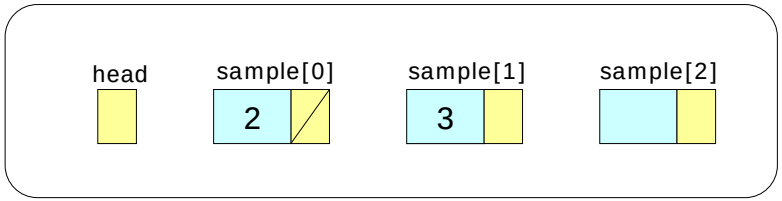
「3」を入力

例題3：解説

Web ページのコーディング例を参照

- 2回目（ i == 1 の場合）

```
scanf("%d", &x );
sample[i].data = x;
```



例題3：解説

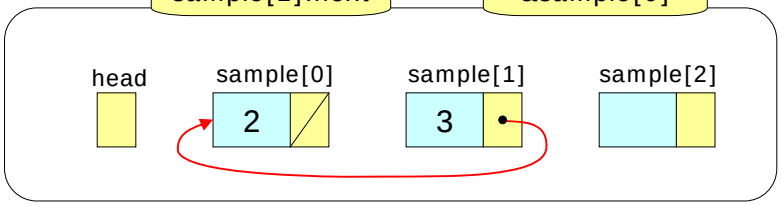
Web ページのコーディング例を参照

- 2回目（ i == 1 の場合）

```
else{
   sample[i].next = &sample[i-1];
}
```

sample[1].next

&sample[0]

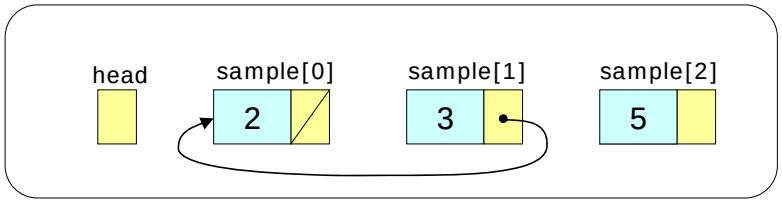


例題3：解説

Web ページのコーディング例を参照

- 3回目（ i == 2 の場合）

```
scanf("%d", &x );
sample[i].data = x;
```



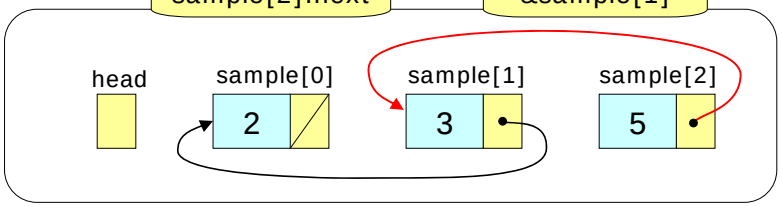
例題3：解説

Web ページのコーディング例を参照

- 3回目（ i == 2 の場合）

```
else{
   sample[i].next = &sample[i-1];
}
```

sample[2].next　　&sample[1]

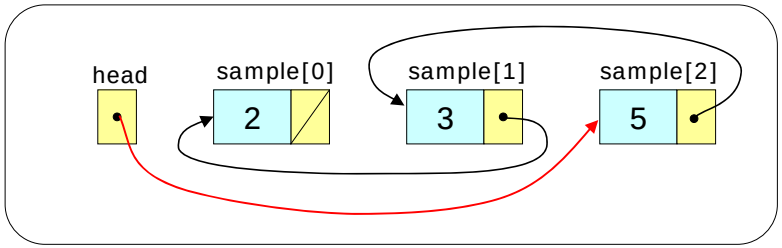


例題3：解説

Web ページのコーディング例を参照

- for 文による繰り返しが終わり，最後に先頭（head）のリンクを形成する

```
head = &sample[2];
```



入力と逆順でのデータ管理

- 例題3の構造では，先に入力されたデータほど後方に格納されることになる
- これを LIFO（ Last-In, First-Out ）といい，多くの場合，**スタック**（stack）という
- 計算機の構成，オペレーティングシステムの設計，各種アプリケーション開発でスタックは非常に重要
  - 詳しくは後期の「データ構造とアルゴリズム」で