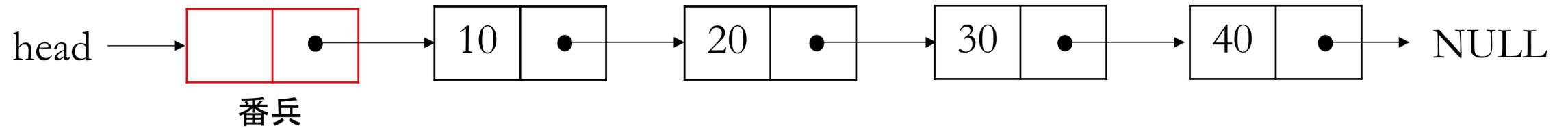


# 連結リスト

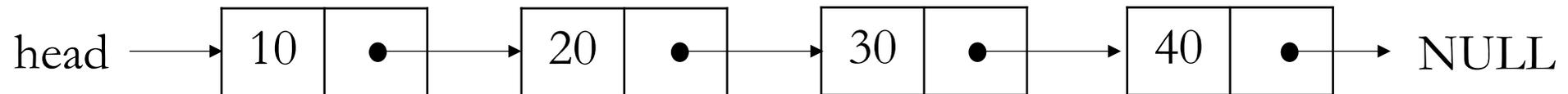
実装-連結リストの作成

# 連結リストの実装は二通り

- 連結リスト(番兵あり)



- 連結リスト(番兵なし)



# 連結リストの実装は二通り

- 連結リスト(番兵あり)
  - 番兵はダミーノードでデータ部にデータを格納しない
  - プログラミングの難易度が低い
  - リストへの削除・挿入操作が直感的でわかりやすい
  - 番兵がなければ削除・挿入において先頭と先頭でないを区別しなければならない。または、下記のようなポインタのポインタを使わなければならない
- 連結リスト(番兵なし)
  - プログラミングの難易度が高い
  - リストへの削除・挿入操作においてポインタのポインタを使う
  - 以降、番兵ありと比較しないとき、単純に「連結リスト」と呼ぶ

# 連結リストの作成

# 二通りの方法が考えられる

1. リストのノードを1つ1つ作成して行く方法
  2. リストへの挿入操作を0から繰り返すことで作成していく方法
- 本授業では連結リストの基本を押さえるために方法1について説明する

# 作成の手順

1. ノードを1つ作成 (malloc関数で動的確保)。最初のノードであればこれをheadで参照できるようにする
2. ノードのデータ部にデータを格納
3. 次のノードを作成
4. 前に作成したノードのポインタ部に、今作成したノードのアドレスを格納
5. すべてのノードが作成されたまで1-4を繰り返す
6. 最後のノードのポインタ部にNULLを与える

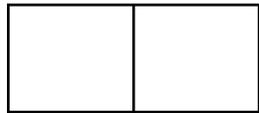
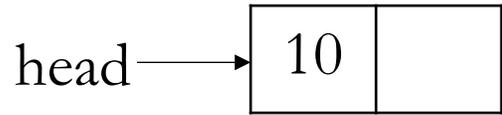
# 作成のイメージ (例: 10 → 20 → 30)



# 作成のイメージ (例: 10 → 20 → 30)

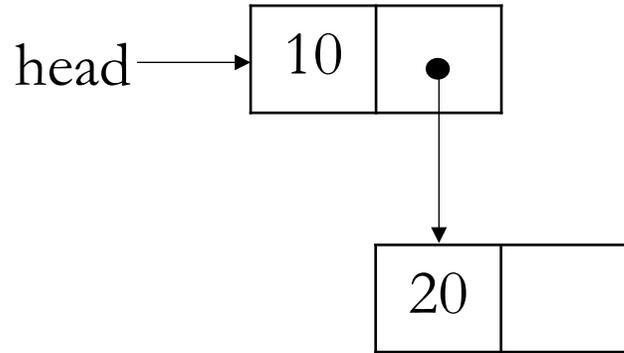


# 作成のイメージ (例: 10 → 20 → 30)



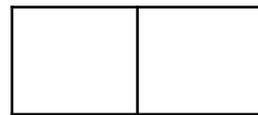
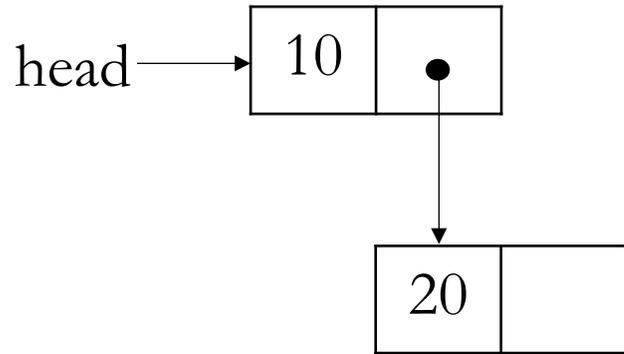
ノードを作成 (malloc関数でメモリ確保)

# 作成のイメージ (例: 10 → 20 → 30)



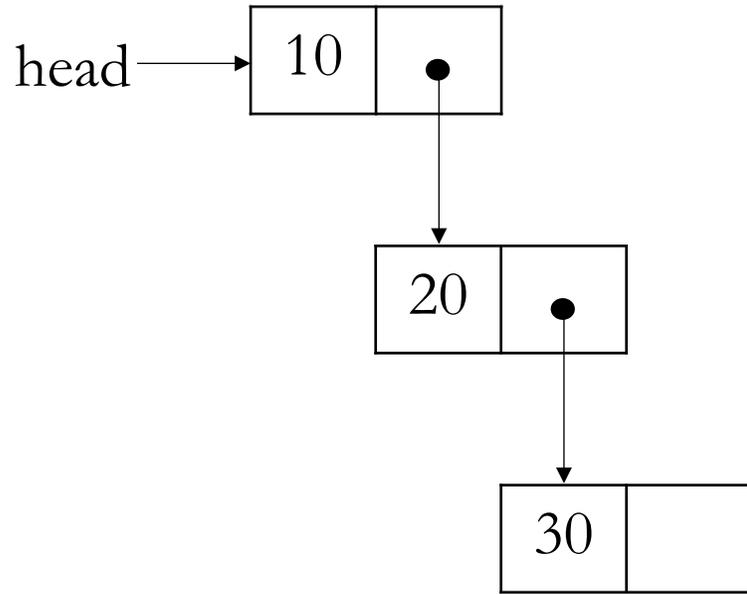
データ部にデータを。ノードのアドレスを前のノードのポインタ部に

# 作成のイメージ (例: 10 → 20 → 30)



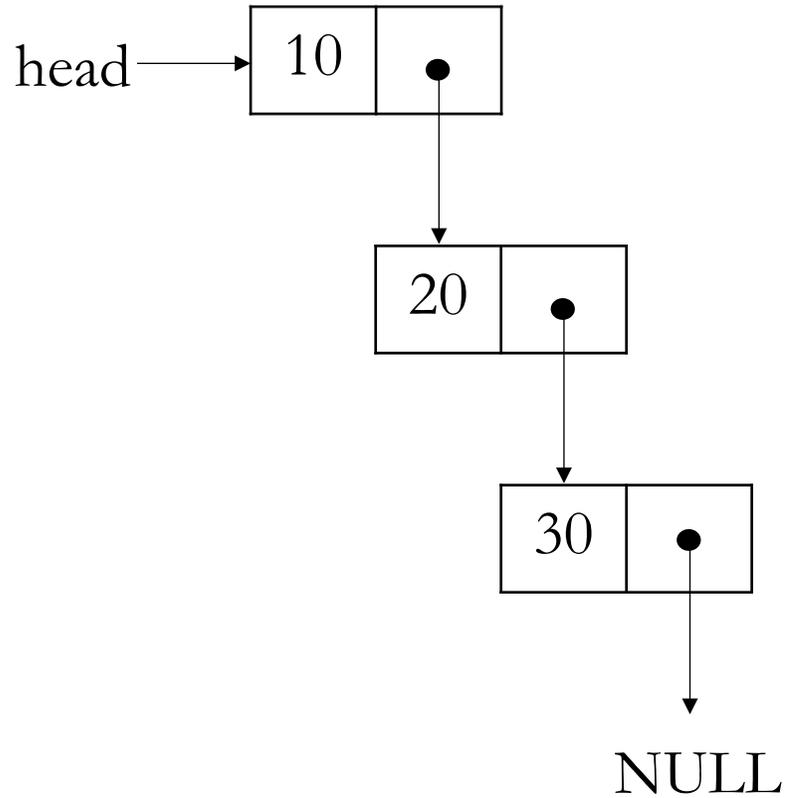
ノードを作成 (malloc関数でメモリ確保)

# 作成のイメージ (例: 10→20→30)



データ部にデータを。ノードのアドレスを  
前のノードのポインタ部に

# 作成のイメージ (例: 10→20→30)

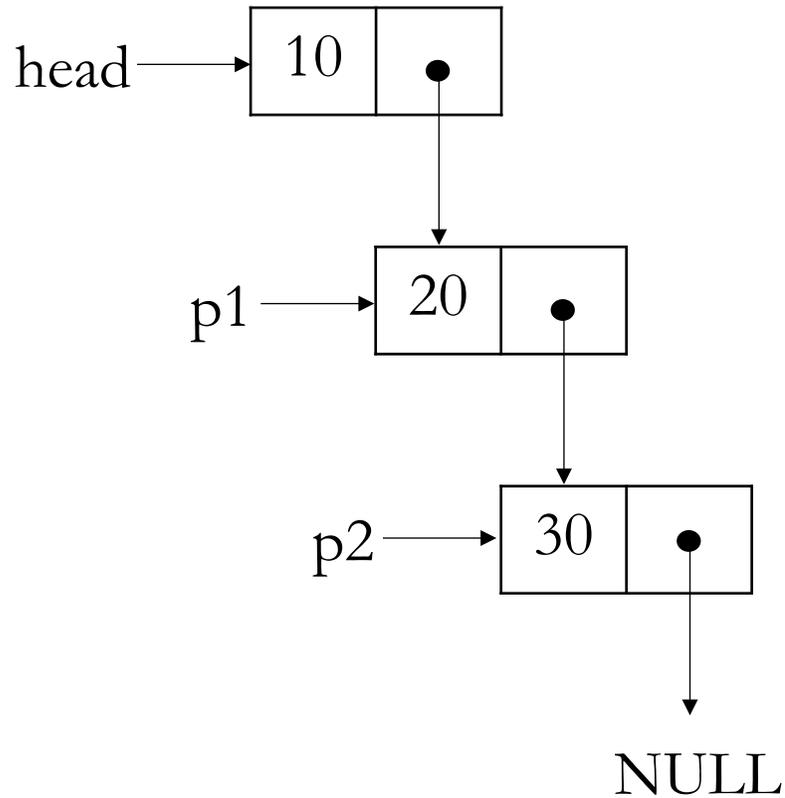


ポインタ部にNULLを

# Question

- このように作成された連結リストは「番兵あり」？「番兵なし」？

# 実装してみる



//ノードの定義

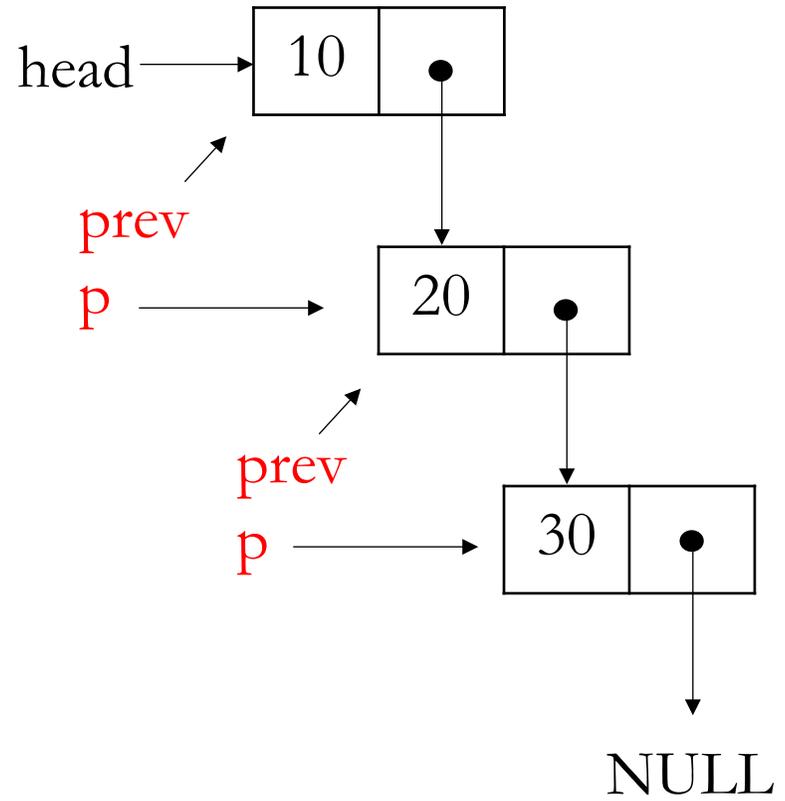
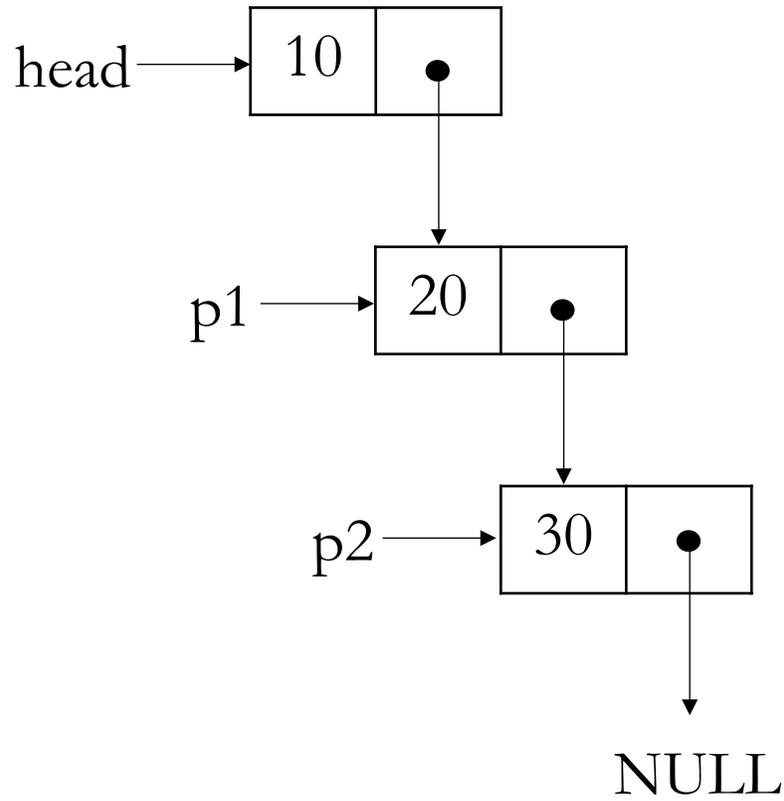
```
typedef int data_t;  
  
typedef struct node {  
    data_t data;  
    struct node *next;  
} NODE;
```

```
NODE *ListMake(void)  
{  
    NODE *head, *p1, *p2;  
    head=malloc(sizeof(NODE));  
    head->data=10;  
    p1=malloc(sizeof(NODE));  
    p1->data=20;  
    head->next=p1;  
    p2=malloc(sizeof(NODE));  
    p2->data=30;  
    p1->next=p2;  
    p2->next=NULL;  
    return head;  
}
```

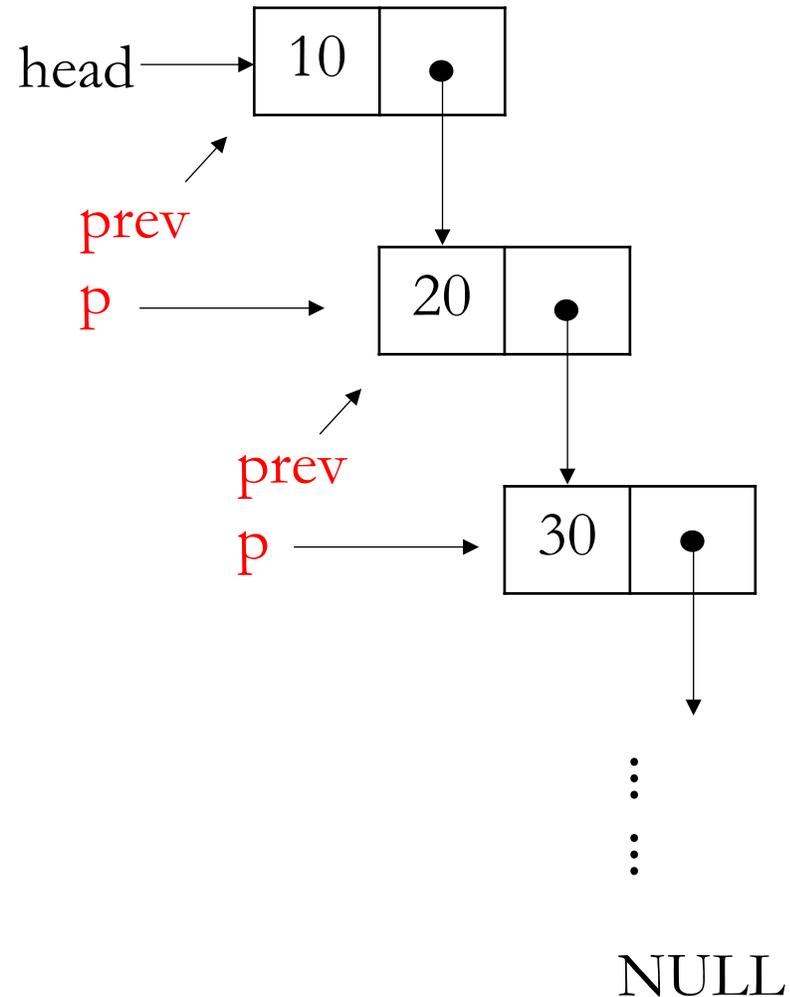
# Question

- 上記実装方法って何が問題？

# 2個の変数でボタンタッチしながら進めていく



# 作成の関数



```
NODE *ListMake(int n, int a[])
{
    NODE *head, *prev, *p;
    prev=head=malloc(sizeof(NODE));
    prev->data=a[0];
    for(int i=1; i<n; i++) {
        p=malloc(sizeof(NODE));
        p->data=a[i];
        prev->next=p;
        prev=p;
    }
    p->next=NULL;
    return head;
}
```

# Question

- 普通の整数配列aに10,20,30,40を格納したとすれば以下でこれらの内容出力することができる

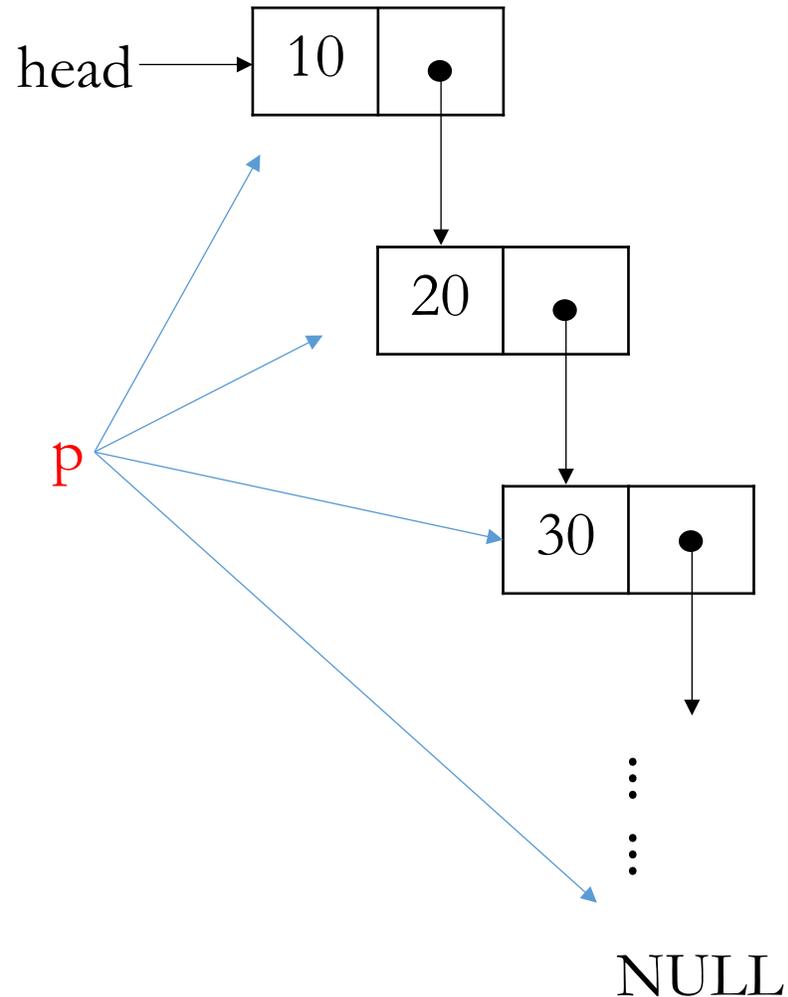
```
int i, a[] = {10, 20, 30, 40};
```

```
for(i=0; i<4; i++) printf(“%d ”, a[i]);
```

```
printf(“¥n”);
```

- さて、ListMake()で連結リスト10,20,30,40を作成したとして、リストの内容をどうすれば出力できるか？

# 連結リストの出力



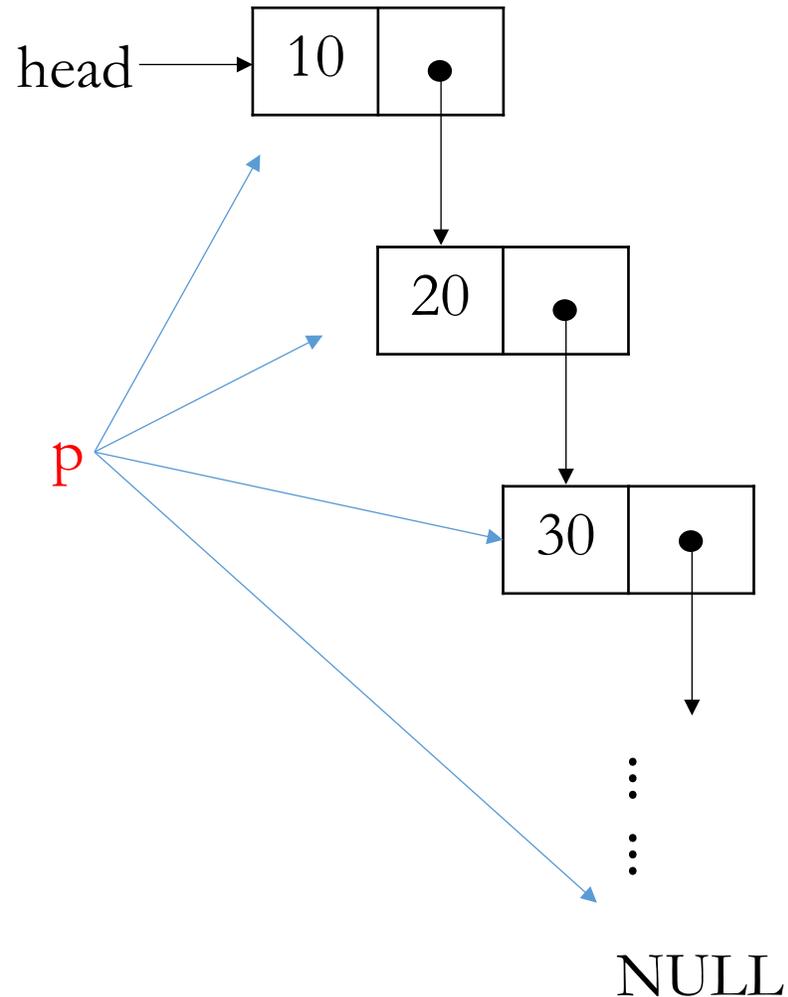
考え方:

pでノードを先頭から指し示して行きデータを順に出力する

手順:

1. ポインタpにheadを与える
2. pがNULLになるまで以下を繰り返す
  - 2.1 pで指し示しているノードのデータを出力
  - 2.2 pをpで指し示しているノードのポインタ部の値で更新

# 出力の関数



ListPrint(head); で呼び出す

```
void ListPrint(NODE *p)
{
    while(p != ?1) {
        printf("%d ", ?2);
        p=?3;
    }
    printf("\n");
}
```

# manaba小テスト:03-1

- 8分
- 6点

# List\*()関数群などをlist.hに格納

- 関数群List\*()は自作のヘッダファイルlist.hに格納されているとする
- また、malloc()関数を使うのに必要な標準ヘッダstdlib.hと、本授業で定義している構造体NODEもlist.hに格納されているとする

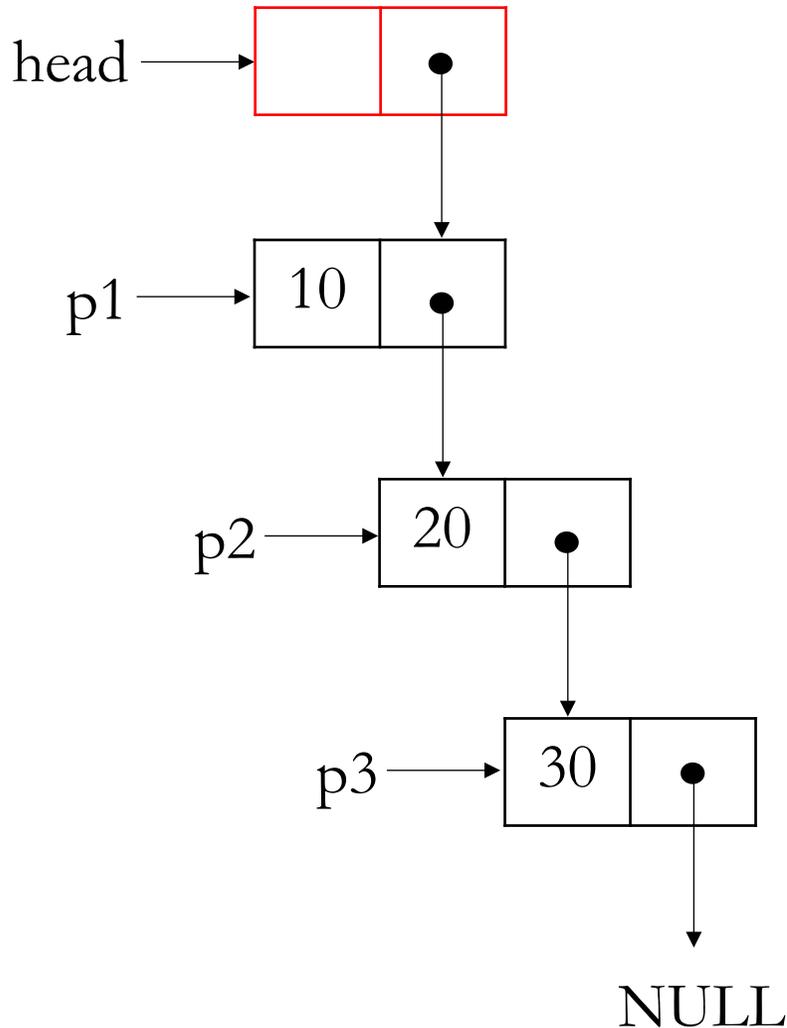
# プログラムを実行してみる

```
#include <stdio.h>
#include "list.h"

int main(void) {
    int n=6, a[]={10, 20, 30, 40, 50, 60};
    NODE *head;
    head=ListMake(n, a);
    ListPrint(head);
    return 0;
}
```

```
./a.out
10 20 30 40 50 60
```

# 番兵ありの作成は？



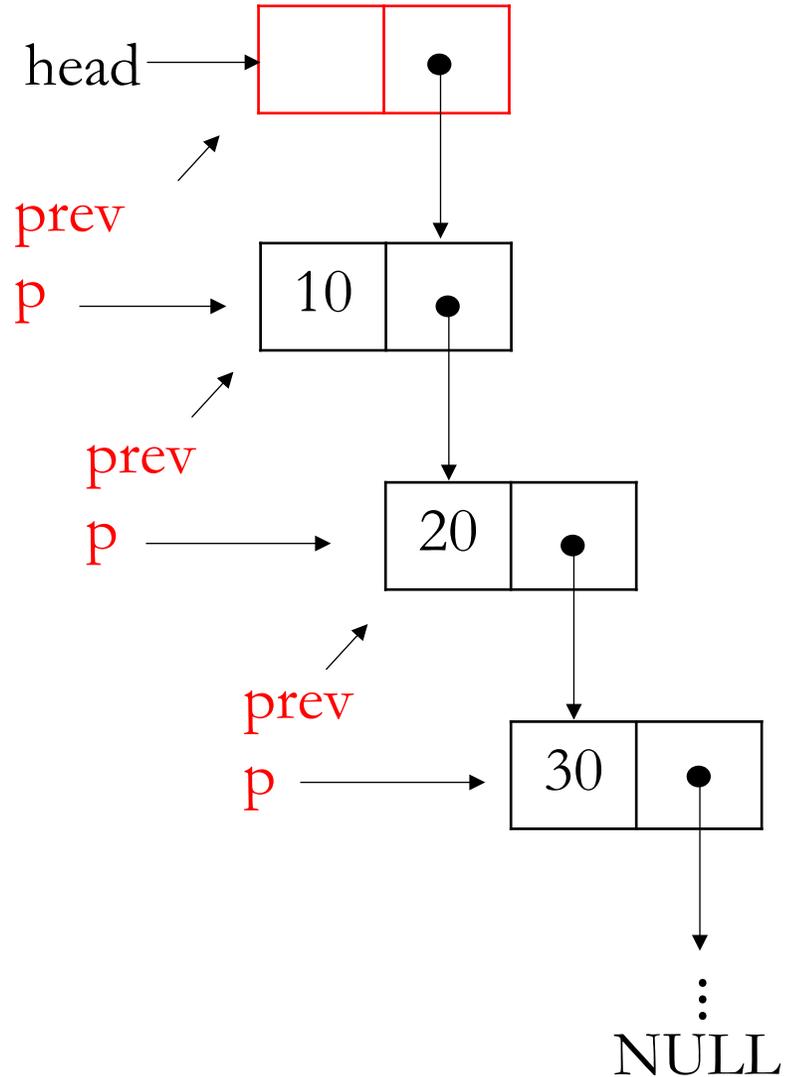
## 「番兵なし」からの修正

```
NODE *ListMake(void)
{
    NODE *head, *p1, *p2, *p3;
    head=malloc(sizeof(NODE));
head->data=10;
    p1=malloc(sizeof(NODE));
    p1->data=2010;
    head->next=p1;
    p2=malloc(sizeof(NODE));
    p2->data=3020;
    p1->next=p2;
p2->next=NULL;
}
```

```
p3=malloc(sizeof(NODE));
p2->next=p3;
p3->next=NULL;

return head;
}
```

# 番兵ありの作成の関数



## 番兵なしからの修正

```
NODE *ListMake(int n, int a[]) {  
    int i;  
    NODE *head, *prev, *p;  
    prev=head=malloc(sizeof(NODE));  
prev->data=a[0];  
    for(i=10; i<n; i++) {  
        p=malloc(sizeof(NODE));  
        p->data=a[i];  
        prev->next=p;  
        prev=p;}  
    p->next=NULL;  
    return head;  
}
```

# 注意

- 番兵使用の場合、ListPrint(head)ではなくListPrint(head->next)で呼び出す必要がある

# manaba小テスト:03-2

- 8分
- 8点