jangjunha <jangjunha113@gmail.com>

C언어 스터디

5/25

오늘의 할 일

- 구조체
- 링크드리스트

구조체

구조체

- Collection of fields

```
struct Person {
    char name[30];
    int age;
    char gender;
};
```

사람에 대한 정보를 나타내는 구조체

구조체

- 여러 사람들의 정보를 저장해봅시다.

```c
#define MAX_PEOPLE 100

char name[MAX_PEOPLE][30];
int age[MAX_PEOPLE];
char gender[MAX_PEOPLE];



strcpy(name[0], "강찬구");
age[0] = 21;
gender[0] = 'M';

strcpy(name[1], "이상혁");
age[1] = 21;
gender[1] = 'M';
```

배열 여러 개로 나타내기

```c
#define MAX_PEOPLE 100

struct Person {
    char name[30];
    int age;
    char gender;
}

Person people[MAX_PEOPLE];

strcpy(people[0].name, "강찬구");
people[0].age = 21;
people[0].gender = 'M';

strcpy(people[1].name, "이상혁");
people[1].age = 21;
people[1].gender = 'M';
```

구조체를 이용하여 나타내기

구조체 정의

구조체 이름

```
struct Person {
    char name[30];
    int age;
    char gender;
};
```

필드들

변수 선언

```
// 변수 선언
Person person1;

// 배열 선언
Person people[100];
```

```
struct Person {
    char name[30];
    int age;
    char gender;
}
```

구조체 정의부

초기화

```
Person person1 = {"김영희", 30, 'F'};
```

필드 정의 순서대로

```
struct Person {
    char name[30];
    int age;
    char gender;
}
```

구조체 정의부

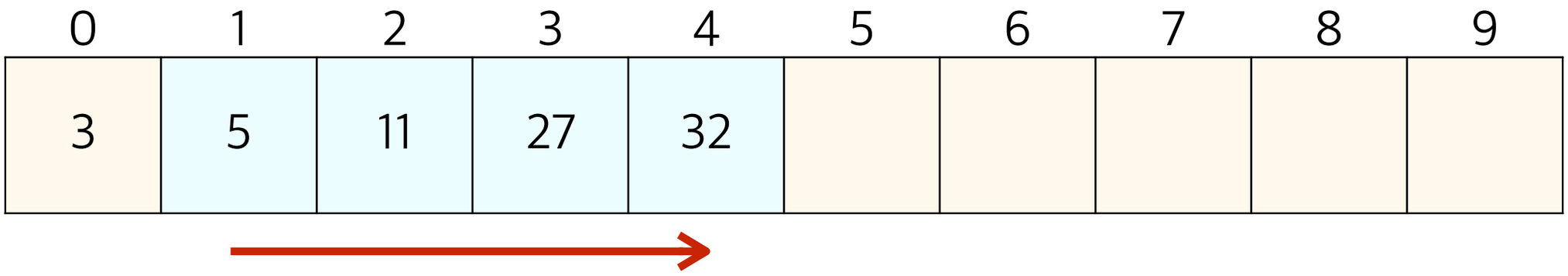
접근

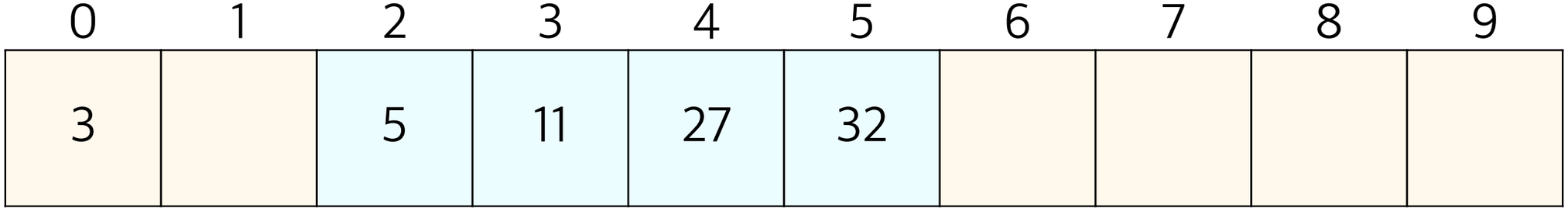
```
Person person1 = {"김영희", 30, 'F'};

printf("이름: %s\n나이: %d\n성별: %c\n\n",
       person1.name, person1.age, person1.gender);

person1.age++;
```

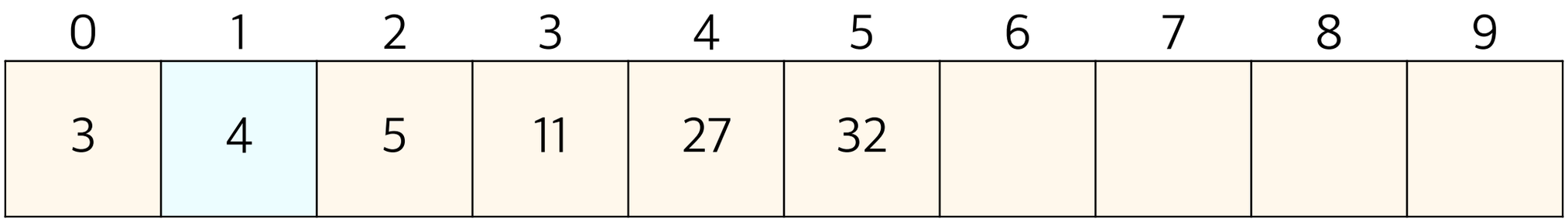
포인터를 써보자

— 링크드리스트 —

배열에서의 원소 삽입


0 1 2 3 4 5 6 7 8 9
3 5 11 27 32


0 1 2 3 4 5 6 7 8 9
3 5 11 27 32


0 1 2 3 4 5 6 7 8 9
3 4 5 11 27 32

배열에서의 원소 삭제

0	1	2	3	4	5	6	7	8	9
3	4	5	11	27	32				

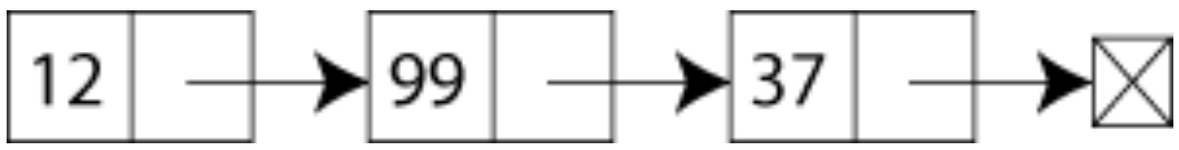
0	1	2	3	4	5	6	7	8	9
3	4	5		27	32				

0	1	2	3	4	5	6	7	8	9
3	4	5	27	32					

링크드리스트

- **연결 리스트, 링크드 리스트(linked list)**

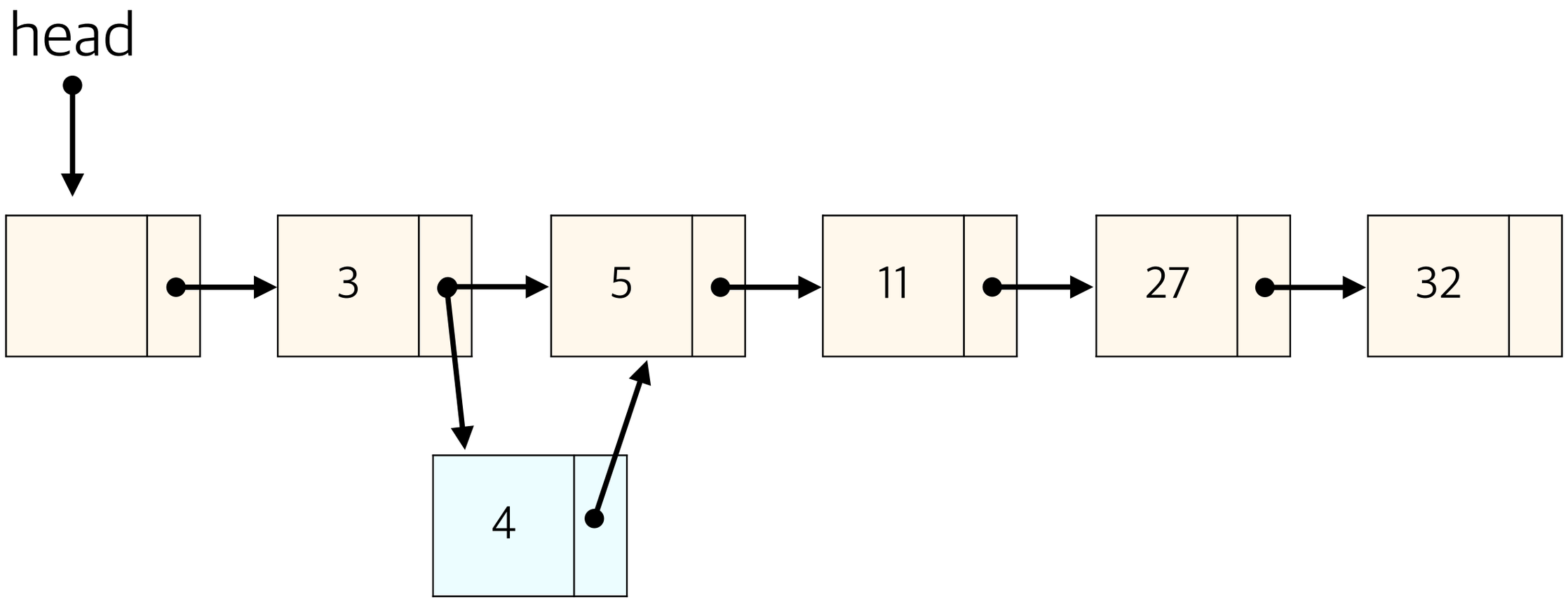
- 각 노드가 데이터와 포인터를 가지고 한 줄로 연결되어 있는 방식으로 데이터를 저장하는 자료 구조



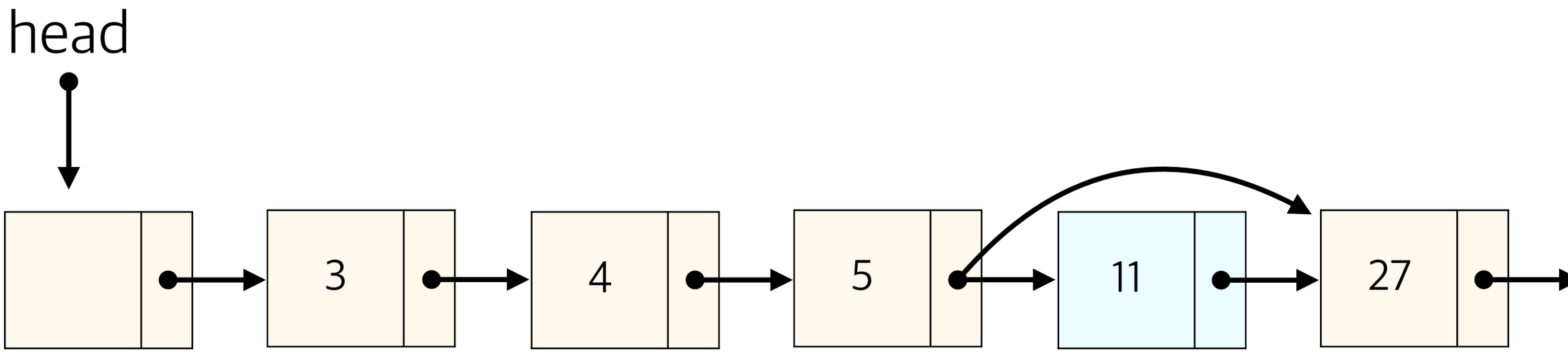
3개의 정수를 저장하고 있는 단순연결리스트

https://ko.wikipedia.org/wiki/연결_리스트

연결리스트에서의 원소 삽입



연결리스트에서의 원소 삭제



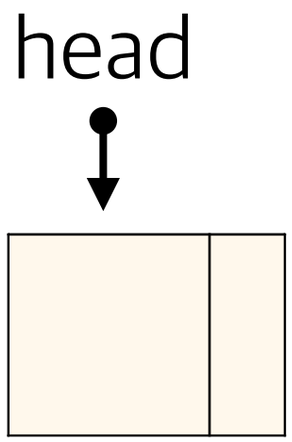
구조체 정의

```
struct Node {
    int value;
    Node *next;
};
```

정수형 데이터 하나를 저장하는 노드

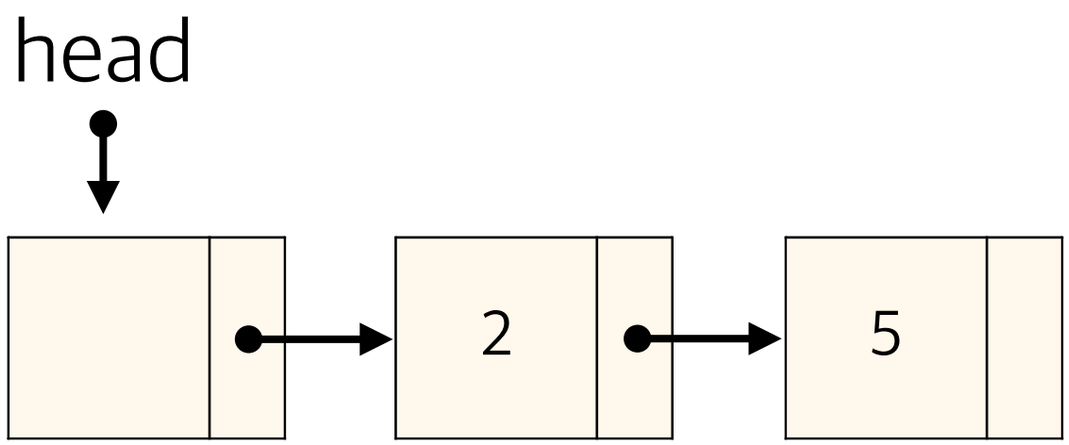
연결리스트 선언

```
Node *head = (Node *)malloc(sizeof(Node))
```



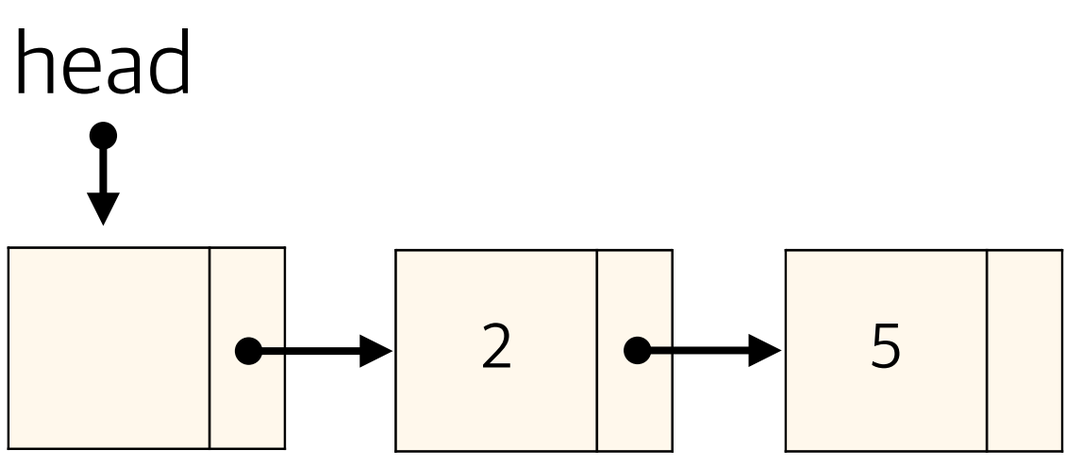
원소 추가

```
Node *head = (Node *)malloc(sizeof(Node));

Node *node1 = (Node *)malloc(sizeof(Node))
node1->value = 2;
node1->next = NULL;
head->next = node1;

Node *node2 = (Node *)malloc(sizeof(Node));
node2->value = 5;
node2->next = NULL;
node1->next = node2:
```



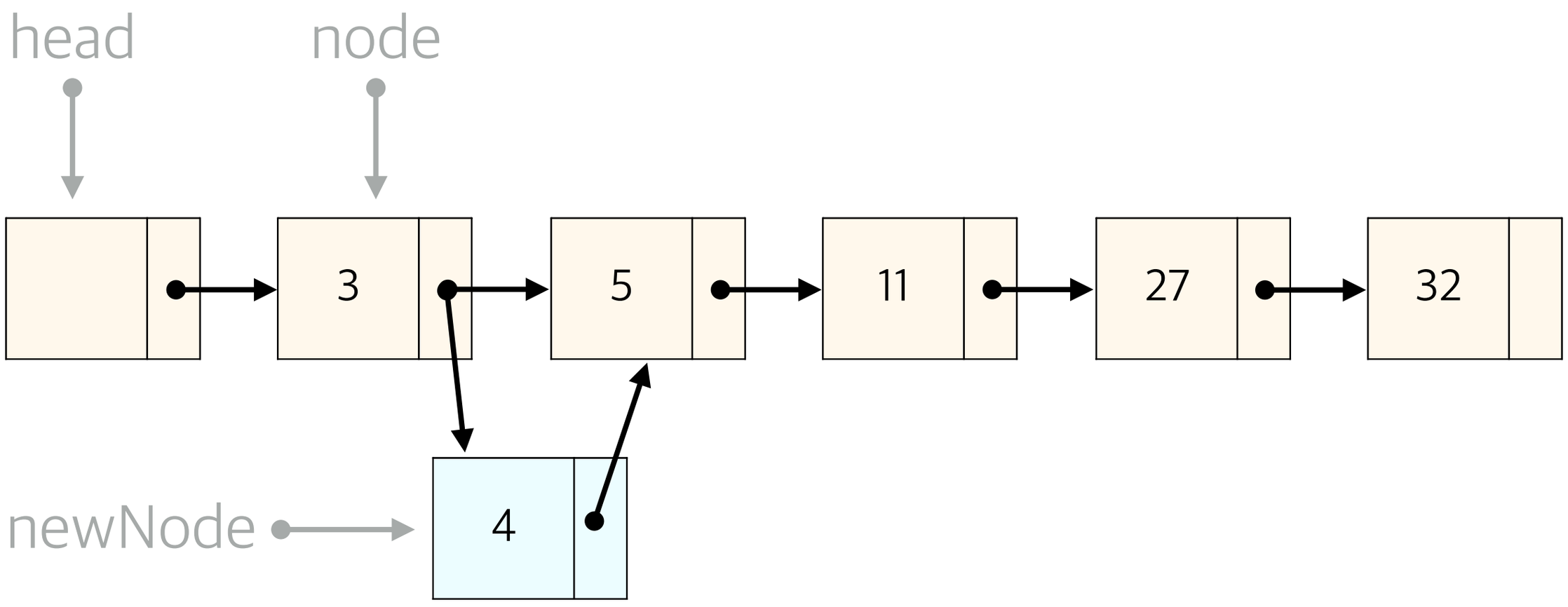
원소 제거

```
node1->next = NULL;
free(node2);
```



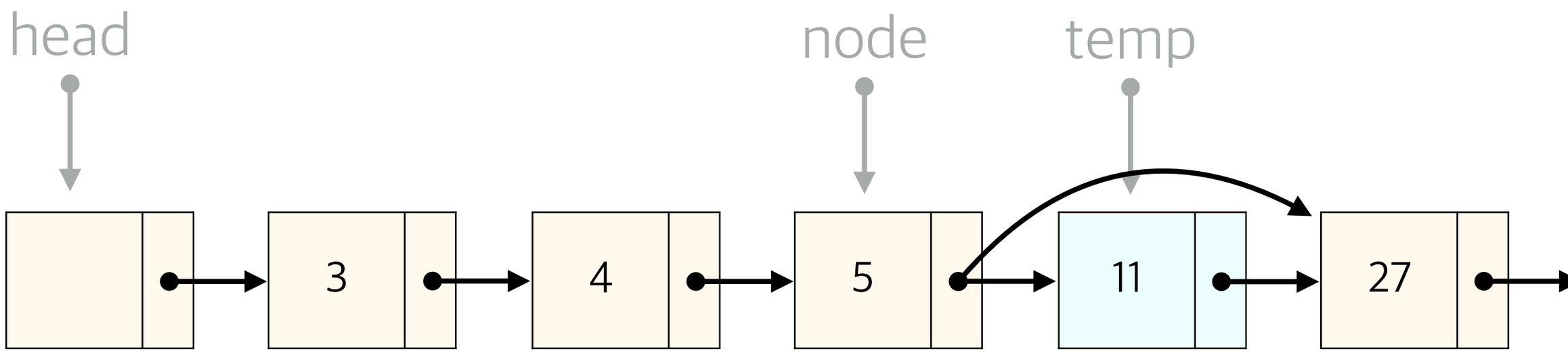
원소 삽입

```
void insertAfter(Node *node, Node *newNode) {
    newNode->next = node->next;
    node->next = newNode;
}
```



원소 삭제

```
void removeAfter(Node *node) {
    Node *temp = node->next;
    node->next = node->next->next;
    free(temp);
}
```

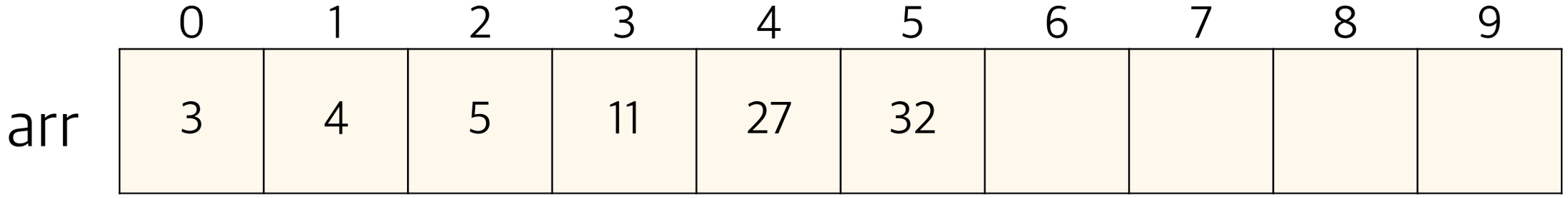


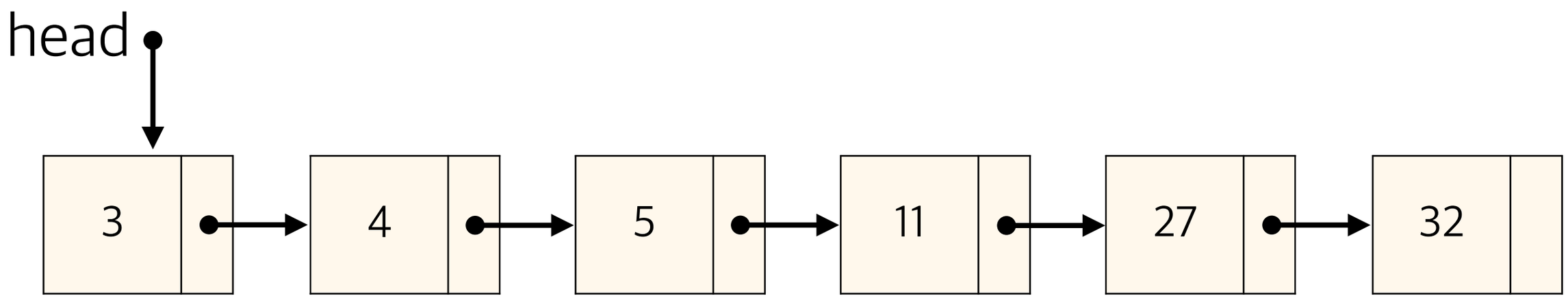
배열 vs 연결리스트

	배열	연결리스트
n번째 원소에 접근	빠름 O(1)	느림 O(n)
원소 삽입·삭제	느림	빠름 O(1)
메모리 사용량	적음	많음

접근

- 3번째 원소를 출력해봅시다.





접근

- 3번째 원소를 출력해봅시다.

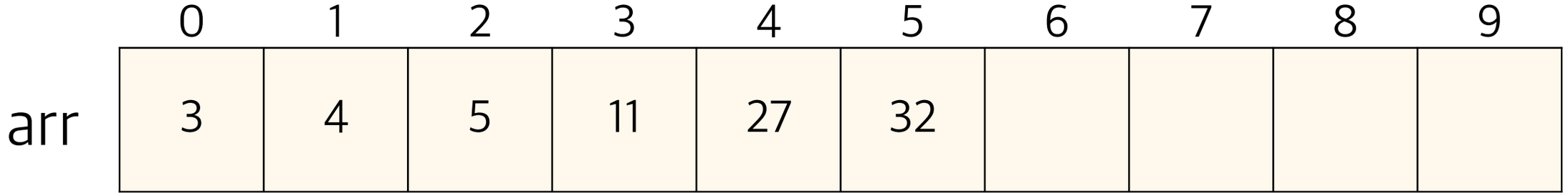
```
printf("%d\n", arr[2]);
```

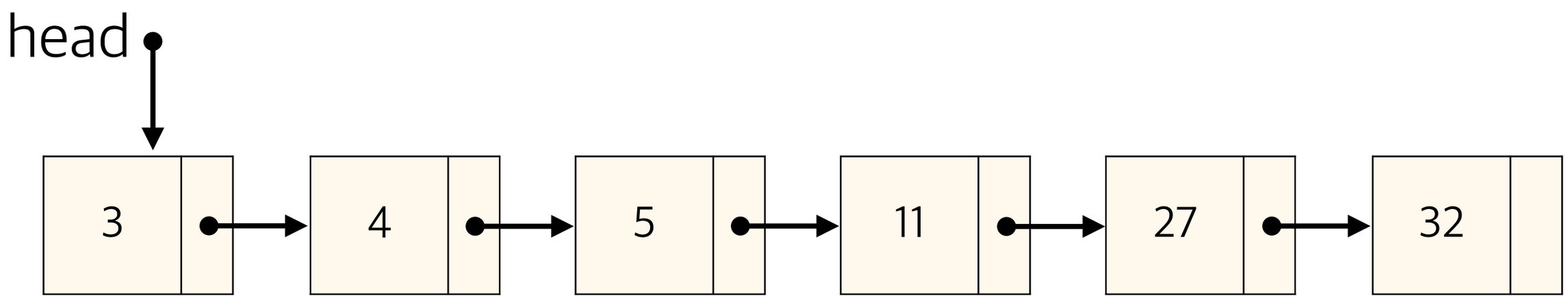
```
void printValue(Node *head, int n) {
    Node *i = head->next;
    while (--n) {
        i = i->next;
    }
    printf("%d\n", i->value);
}
```

순회

- 모든 원소를 출력해봅시다.

```
3  4  5  11  27  32
```





순회

- 모든 원소를 출력해봅시다.

```
3  4  5  11  27  32
```

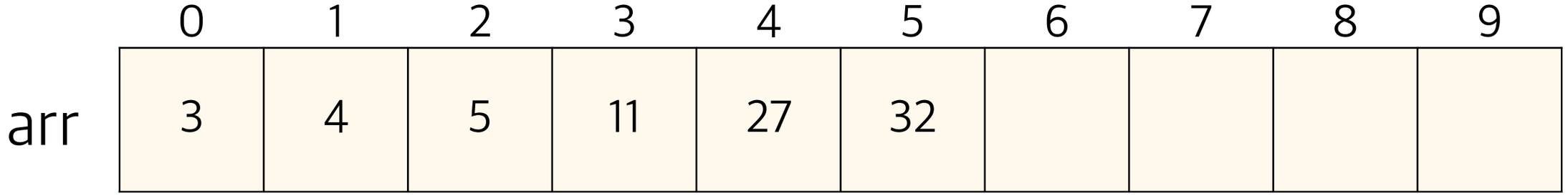
```
for (int i = 0; i < 6; i++)
     printf("%d ", arr[i]);
```

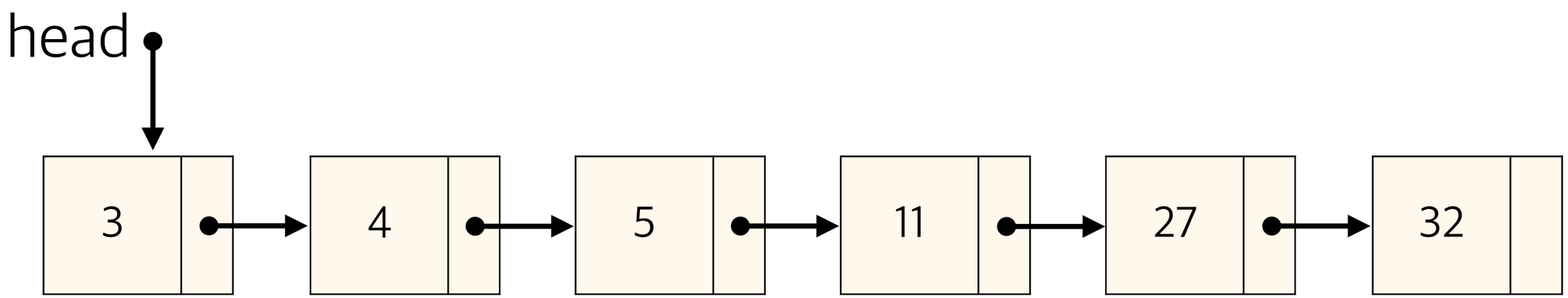
```
void printList(Node *head) {
   Node *i;
   for (i = head->next; i; i = i->next) {
      printf("%d ", i->value);
   }
   printf("\n");
}
```

역방향 순회

- 모든 원소를 역순으로 출력해봅시다.

```
32 27 11 5 4 3
```





끝.