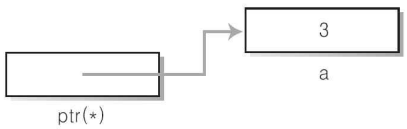
# 10_포인터 타입

- 포인터의 개요
  - 포인터 타입
    - 변수가 임의의 객체를 참조하기 위해 메모리 주소를 값으로 갖는 타입
    - Pascal, C, C++, Ada 등 최근 언어들은 포인터 개념을 제공
  - 포인터 개념이 도입된 큰 이유
    - 기억장소의 동적 관리
    - 동적으로 할당받는 메모리 공간 : heap
  - 포인터 개념

```
01  int a=3;
02  int *ptr;
03  ptr = &a;
04  printf("%d", *ptr);
```




# 10_포인터 타입

- 포인터 변수 선언 시, 변수 이름 앞에 * 를 붙임
- ptr : 포인터 변수
  - int 타입의 데이터를 저장하고 있는 메모리 주소를 저장
  - 포인터 변수에 *를 붙이면, ptr이 가리키는 곳을 의미

- 포인터를 이용한 동적 기억 장소 관리
  - 예: 구조체 선언, 포인터 타입 변수 선언 후 동적 메모리 할당

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

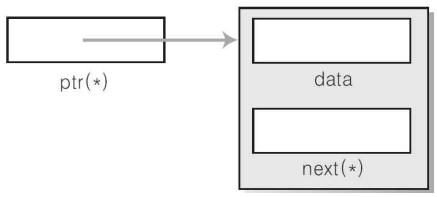
```
struct list *ptr;
```

  - 생성되는 ptr은 struct list 타입 데이터를 저장하고 있는 메모리 주소를 저장할 수 있음
  - malloc 함수 사용, 동적으로 메모리를 할당한 후 ptr이 이 영역을 가리킴

# 10_포인터 타입

```
ptr = (struct list *)malloc(sizeof(struct list));
```

- malloc 함수 사용, 동적으로 메모리를 할당한 후 ptr이 이 영역을 가리킴

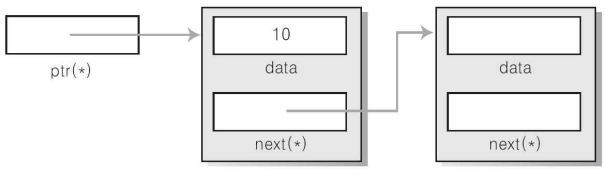



- ptr이 가리키는 영역의 data에 접근하는 표현 형식 : ptr->data
  - data에 10을 저장하는 방법 : ptr->data = 10;

# 10_포인터 타입

- 또 다른 메모리 영역을 동적으로 할당받고 ptr->next가 이 영역을 가리키게 하는 코드

```
ptr->next = (struct list *)malloc(sizeof(struct list));
```




- 새롭게 생성된 영역의 data에 20 저장

```
ptr->next->data = 20;
```

- 다른 영역을 가리키고 있지 않음을 의미하는 NULL을 저장

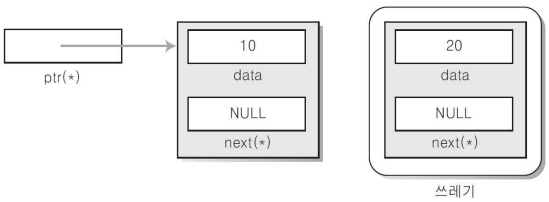
```
ptr->next->next = NULL;
```

## 10_포인터 타입

- 이 상태에서 다음을 실행

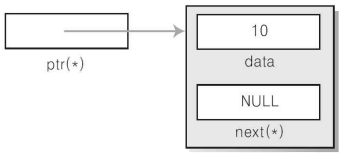
```
ptr->next = NULL;
```

- 쓰레기 발생 : 20을 저장한 영역은 더 이상 접근 불가




- 동적으로 할당된 영역을 회수하기 위한 free 함수

```
free(ptr->next);
ptr->next = NULL;
```




## 10_포인터 타입

- 참조 타입
  - C++는 참조 타입이라는 포인터 타입을 추가적으로 제공
    - 참조 타입 변수는 변수 선언과 동시에 반드시 초기화되어야 함
    - 그 후에는 값을 변경할 수 없음
    - 참조 타입 변수는 선언할 때 변수 이름 앞에 &를 붙임
    - C++ 예

```
01  int val = 10;
02  int &ref = val;
      :
03  ref = 20;
```

      - 참조타입 변수 ref 선언 후, val과 ref는 이름만 다른 같은 변수가 됨
      - 3행 실행 후 val도 20이 됨

# 10_포인터 타입

- 부프로그램의 형식 매개 변수로 사용할 때 매우 유용
  - 포인터를 이용해서 main의 count 변수의 값을 1 증가시키는 예

```
void incr (int *ptr)
{
  (*ptr)++;
}
int main (void)
{
   ⋮
  incr (&count);
   ⋮
}
```