동적 메모리의 소유권 문제

2021년 2월 17일 수요일 오후 10:34

237. 동적 메모리의 소유권 문제

- 동적으로 할당한 메모리의 큰 문제
  - 근데 큰 문제가 하나 있음
  - 동적으로 할당한 메모리의 소유주는 누구인가?
  - 바로, 메모리를 생성한 함수
  - 소유주란? **그 메모리를 반드시 책임지고 해제해야 하는 주체**
  - 소유주가 아닐 때는 그냥 빌려 사용할 뿐 해제하면 안 됨!
    - 소유주와 비소유자가 모두 해제하면 문제
    - 근데, 한 명도 해제 안 해도 문제
- 이 문제의 예
  - 호출자가 이 함수를 호출하는 순간 내부에서 새 메모리가 할당해서 반환한다는 사실을 어떻게 알지?

```
const char* combine_string(const char* a, const char* b)
{
    void* str;
    char* p;
    /* a와 b의 길이 및 두 길이를 더한 값을 보관한 변수 생략 */

    str = malloc(size);

    /* a와 b를 str에 복사하는 코드 생략*/

    return str;
}
```

```
result = combine_string("Hello", "World");
```

    - 문자열을 반환하는데 그 문자열이 동적으로 할당한 건지 알 수가 없다
- C++에서는 RAII로 해결
  - 자원 획득은 초기화(RAII, Resouce Acquisition Is Initialization)
    - 여기서 자원은 메모리
  - C++은 개체지향을 지원하는 언매니지드 언어
  - 한 개체가 생성될 때 필요한 메모리를 할당
    - 생성자란 특별한 함수
  - 그 개체의 수명이 다할 때 그 메모리를 해제
    - 소멸자란 특별한 함수
  - 즉, 개체의 수명이라는 범위에 메모리의 수명을 종속시킴
  - 이 원칙을 잘 따르면 실수할 여지가 적음
- 하지만 C는 개체가 없다!!
  - 그래도 RAII를 최대한 흉내 내면 좋음
  - C에서 RAII를 할 수 있는 부분
    - 한 함수 안에서 malloc(), free()를 다 호출할 수 있는 경우
  - 이런 이유 때문에 저 앞에 예에서 malloc()을 추가하면 곧바로 free()를 추가하라고 한 것임

```c
void do_something(void)
{
    void* nums;

    nums = malloc(10 * sizeof(int));
    free(nums); /* 일단 넣고 함수 구현할 것! */
}
```

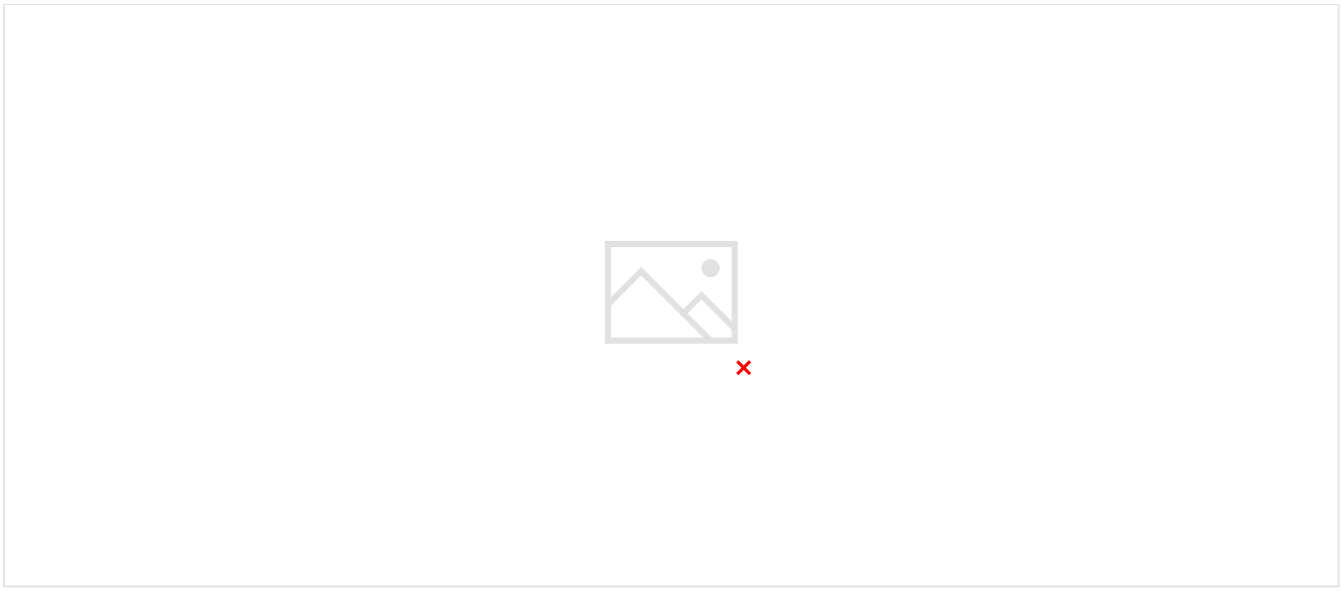
- 하지만 중간에 함수를 탈출할 경우..
  - 코드 중간에 return 하면 말짱 도루묵

```c
void do_something()
{
    void* nums;

    nums = malloc(10 * sizeof(int));

    /* 코드 생략 */

    if (조건) {
        return; /* free(nums)를 못 함!!! */
    }

    free(nums);
}
```

- 해결법: goto 문을 사용

```c
void do_something()
{
    void* nums = NULL;

    nums = malloc(10 * sizeof(int));

    /* 코드 생략 */

    if (조건) {
        goto free_and_exit;
    }

    /* 코드 생략 */

free_and_exit:
    free(nums);
}
```

  - malloc 안하고 free 해버리는 경우를 방지하기 위해서 nums를 NULL로 초기화
- 그럼 원래의 문제는 어떻게 해결할까?
  - 함수에서 메모리 동적 할당 한 다음에 free를 못하는 경우
  - C에서 굉장히 해결이 어려운 문제
  - 최선의 방법은?
  - **이런 함수가 최대한 없게 한다**
  - combine_string() 예의 경우, 함수 안에서 할당하는 대신 함수 밖에서 할당 후 매개변수로 전달
- 간단히 보는 combine_string() 해결법



    - 길이를 구하는 함수로 두 문자열을 합친 길이를 구하고
    - 그 길이만큼 할당을 함
    - 그리고 본 함수에다가 매개변수로 전달
- 동적으로 할당 후 반환을 피할 수 없다면?
    - 딱히 모두가 동의하는 표준은 없음
    - 어떤 함수가 메모리 할당을 한다면 함수에 그 사실을 표기
        - 주석으로 표기하는 법도 있음. 그러나 사람들은 주석을 잘 안 읽음
    - 동적 메모리를 할당하는 변수라면 변수명에 표기하는 법도 있음
- 코딩 표준: 동적 메모리 할당을 하는 함수명
    - 어떤 함수의 내부에서 동적 메모리 할당을 한다면, 이름으로 확실히 알려주자!

- 코딩 표준: 동적 할당 메모리 저장 변수명
    - 동적 할당된 메모리를 저장하는 변수라면, 이름에 그 사실을 명시
        - pa: pointer allocated

- 베스트 프랙티스 정리

1. **malloc() 작성한 뒤에 곧바로 free()도 추가하자**
    - free()를 절대 까먹으면 안 됨!!
2. **동적 할당을 한 메모리 주소를 저장하는 포인터 변수와 포인터 연산에 사용하는 포인터 변수를 분리해 사용하자**
    - 원래 포인터 변수를 사용할 경우, 주소를 잃어버려서 해제를 못할 수 있음

3. **메모리 해제 후, 널 포인터를 대입하자**
    - free()를 두 번 호출하는 문제를 방지하기 위함

4. **정적 메모리를 우선으로 사용하고 어쩔 수 없을 때만 동적 메모리를 사용하자**
5. **동적 메모리 할당을 할 경우, 변수와 함수 이름에 그 사실을 알리자**

    - 함수 안에서 할당하고 반환하는 패턴은 최대한 피하자