Day-24

Heap Memory - new, delete Keywords

new, delete

- new/new[] 및 delete/delete[]는 런타임에 동적 메모리 할당/해제 및 객체 생성/삭제를 수행하는 C++ 키워드
- malloc과 free는 C함수, 메모리 블록(바이트 단위)만 할당하고 해제함

new, delete Advantages

- 함수가 아닌 키워드이기 때문에 더 안전함
- Retrun type
  - new는 정확한 데이터 유형을 반환
  - malloc()은 void*를 반환
- Failure
  - new는 예외를 throw함
  - malloc()은 NULL 포인터를 반환 → 무시할 수 없음
  - 0 크기 할당은 특별한 코드가 필요하지 않음
- 할당 크기
  - new키워드를 사용하여 컴파일러가 바이트 수를 계산
  - malloc()은 크기를 수동으로 계산
- 초기화
  - new를 사용하여 할당 외에도 초기화 가능
- 다형성
  - 가상 함수가 있는 객체는 가상 테이블 포인터를 초기화하기 위해 new로 할당해야 함

동적 메모리 할당

- 단일 요소 할당

```
int* value = (int*) malloc(sizeof(int)); // C
int* value = new int;                    // C++
```

- N개의 요소 할당

```
int* array = (int*) malloc(N * sizeof(int)); // C
int* array = new int[N];                     // C++
```

- N개의 구조체 할당

```
MyStruct* array = (MyStruct*) malloc(N * sizeof(Mystruct))
MyStruct* array = new Mystruct[N];
```

- N요소 할당 및 0 초기화

```
int* array = (int*) calloc(N, sizeof(int)); // C
int* array = new int[N]();                  // C++
```

동적 메모리 할당 해제

- 단일 요소 할당 해제

```
int* value = (int*) malloc(sizeof(int)); // C
free(value);

int* value = new int;                    // C++
delete value;
```

- N개 요소 할당 해제

```
int* array= (int*) malloc(N * sizeof(int)); // C
free(array);
```

```
int* array = new int[N];                    // C++
delete[] value;
```

할당/할당해제 속성

- 기본 규칙
  1. malloc()으로 할당된 각 객체는 free()로 할당 해제해야 함
  2. new로 할당된 각 객체는 delete로 할당 해제해야 함
  3. new[]로 할당된 각 객첸느 delete[]로 할당 해제해야 함
  4. malloc, new, new[]는 0 크기 할당(구현 정의)를 제외하고 성공할 경우 NULL 포인터 생성 X
  5. free(), delete, delete[]를 NULL/nullptr 포인터에 적용해도 에러 생성X
- new, new[], malloc을 다른 것과 혼합해서 사용하면 정의되지 않은 동작 발생